*Fundamenta Informaticae 123 (2010) 1–12*
*DOI 10.3233/FI-2010-330*
*IOS Press*

# Modeling Biology using Generic Reactive Animation

**Yaki Setty**[*]
*Computational Science Lab*
*Microsoft Research, Cambridge, UK*
*yaki.setty@gmail.com*

**Irun R. Cohen**
*Department of Immunology*
*The Weizmann Institute of Science*
*Rehovot, Israel*

**David Harel**
*Department of Computer Science*
*The Weizmann Institute of Science*
*Rehovot, Israel*

**Abstract.** Complex biological systems involve incorporated behaviors of numerous processes, mechanisms and objects. However, experimental analysis, by its nature, divides biological systems into static interactions with little dynamics. To bridge the gap between experimental data and the underlying behavior, our group has been formalizing biological findings into mathematically and algorithmically rigorous specifications, which are then compiled into reactive models. To realistically animate our models, we designed a generic architecture for the earlier idea of reactive animation, in a way that allows it to link up reactive models with animation tools. Here, we describe the reactive animation approach and some of the benefits of employing it to simulate and analyze complex biological systems. We illustrate our approach with a model of pancreatic development, a highly complex system with a unique 3D structure, and also mention more recent work on adding animation to the generic cell project (GemCell).

[*]Address for correspondence: Computational Science Lab, Microsoft Research, Cambridge, UK

[1]This research was supported in part by The John von Neumann Minerva Center for the Development of Reactive Systems, and by a grant from the Kahn Fund for Systems Biology, both at the Weizmann Institute of Science. Part of this David Harel's work carried out during a visit to the School of Informatics at the University of Edinburgh, which was supported by a grant from the EPSRC.

**Keywords:** Modeling, Computational Biology, Reactive Animation

## 1. Introduction

In recent years, various approaches were suggested to amass accumulating biological data in order to help understand experimental results and come up with new insights and hypothesis. Along these lines, our group has been involved for ten years in modeling biological systems. Our approach is guided by the observation that biological systems can be beneficially specified as reactive systems; that is, systems, such as cell phones and ATM machines, that maintain an ongoing interaction with their environment rather than produce some final value upon termination [17, 21]. Accordingly, we have been employing ideas and tools from software engineering to model biological systems as reactive systems, yielding executable, simulate-able models that qualitatively represent biological systems (see e.g., [35, 32, 24]).

One of our observations has been that modeling biological systems as reactive models can be significantly enhanced by realistic interactive animation. Such animation elucidates the behavior of the system by visual representation of the complex interactions in the model, and allows the user to introduce changes in the model, to see the outcome of what-if scenarios, and to test hypotheses in intuitively natural ways. In many cases, the animation reveals behavioral properties that emerge as a consequence of massive execution of basic elements that act in concert as a population [8]. In general, emergent properties are dynamic properties of populations, thus it is rather difficult to predict them from the model's static specifications.

To animate our models, we use a technique, called *reactive animation*, whereby reactive systems are modeled by languages and tools for the reactivity and are linked with tools for realistic animation [11]. Recently, we implemented a generic platform for reactive animation that provides a specific mechanism for linking any number of tools, including reactive engines and various analysis tools [22]. In particular, this architecture supports animation tools (e.g., 3DGameStudio, Flash) and reactive system development environments of various kinds (e.g., Rhapsody, Play-Engine). In this paper, we describe how this generic platform is employed to model biological systems and discuss some of the benefits that result.

Our main example employs this tool to model pancreatic development in the mouse, which in concert with its biological importance, turns out to be a highly complex system for modeling with numerous different kinds of objects. Our model includes a state-based specification linked to a 3-Dimensional animated front-end and a mathematical analysis GUI [33]. A prerecorded clip of the simulation at run-time is available at `www.wisdom.weizmann.ac.il/~yaki/runs`. In this model, Statecharts [16] (in Rhapsody [37]) describe the behavior of the biological objects themselves (e.g., cells), which are represented in the front-end as 3D elements possessing realistic animation attributes. In addition, statistics and analysis of the simulation are shown in a separate GUI (in MATLAB).

The front-end visualizes the animation continuously and provides the means to interact with it. At the same time, the analysis GUI provides statistics and graphs of the simulation. Generally, we found that our simulation of pancreatic organogenesis agrees with the biology, indicating that the emerging 3D structure captures pancreatic morphogenesis in mice. Furthermore, models based on generic reactive animation enable *in silico* experiments at run-time. In pancreatic organogenesis, our case study, the model anticipated results of similar *in-vivo* experiments and suggested new intriguing results. **Currently we are collaborating with developmental biologists, seeking ways to test the predictions experimentally.**

Recently, the generic reactive animation platform was employed to link a relatively new project, the *GemCell* [1] platform, to an animated front-end. The 2D front-end (in Flash) visualizes the generic statechart model of cell behavior, which captures the five main aspects of cell behavior – proliferation, death, movement, import and export [6]. At run-time, the front-end continuously visualizes the life cycle of cells in the GemCell platform. Currently, this project is being tested for modeling various biological processes and the animation helps to disclose the behaviors of elements in the simulations.

## 2. The Modeling Approach

Our entire modeling approach is computational. It is centered around the construction of an executable program that can qualitatively simulate the dynamic behavior of the test biological system over time and under any set of circumstances (from among those that are supported by the model's basic elements, of course). Components of such a simulation represent biological entities, such as cells, which react by various transformations to events involving neighboring components. Computational modeling provides simulations that are effective in many cases where precise quantitative relationships are unknown. Such models can also be constructed when experimental data are incomplete, and they then provide a means to test various hypotheses to fill in the gaps. Analyzing such a model requires mutating behaviors of elements therein and studying their effect on the system. This subject was recently reviewed in several publications that probe different challenges and approaches in computational modeling [13, 9, 31].

We formalize biological finding using the language of Statechart[16], a visual language that is naturally suited for the specification of objects that have clear internal behavior. Together with object model diagrams [19], they provide an object oriented graphical representation of the dynamics of objects using states, transitions, events, and conditions. The Statecharts language makes it possible to visualize the behavior of an object in a way that emphasizes the elements in its life-cycle. Behavior in Statecharts is described using states, and events that cause transitions between states. States may contain substates, thus enabling description at multiple levels, and zooming in and zooming out between levels. States may also be divided into orthogonal states, thus modeling concurrency, and allowing the system to reside simultaneously in several different states. Statecharts are intuitive for users, yet have mathematically well-defined semantics, and are therefore amenable to execution by computers.

Gene activity in statecharts, for example, could be abstracted in a binary fashion to an orthogonal component that consists of two states, `Expressed` and `Unexpressed,` and two transitions, `Promote` and `Inhibit,` that connect them. Once an `Active` event is triggered, the component will change its active state from `Unexpressed` to `Expressed`.
Statecharts can be compiled into executable machine code using various tools, such as Rhapsody by IBM `http://www-01.ibm.com/software/awdtools/rhapsody/`, National Rose by IBM `http://www-01.ibm.com/software/awdtools/developer/rose/`, Matlab `http://www.mathworks.com/` and other. The resulting program is then linked up with an animated front-end, designed using animation and game tools, such as Macromedia Flash `http://www.adobe.com/products/flash/` or 3DGameStudio `http://www.3dgamestudio.com/`. Elements in the front-end correspond to components in the model and certain transitions notify the front-end to carry out property changes and indicate the changes in the model (e.g., cell color change indicates differentiation). Furthermore, the user may interact with the model through the front-end itself by triggering events in the reactive model. For example,

by clicking on the image of a cell, the user queries the model for static as well as run-time information related to that cell.

Initially, reactive animation was utilized to model behavior in two organs: T-cell maturation in the thymus gland [10] and overall developmental dynamics of the lymph node [34]. In both cases, the implementation linked a statechart model of the system (in Rhapsody) to a 2D animation front-end (in Flash), in an ad-hoc fashion. In [22] we described a more generic platform for reactive animation that supplies a specific mechanism for linking any number of tools, in particular animation tools and reactive system engines. The generic reactive animation platform was found general enough to support and maintain various tools of different kinds. A more detailed description of this work, including examples and interactive illustrations, can be found at `www.wisdom.weizmann.ac.il/~yaki/GRA`.

Computational approaches such as the one suggested here, require taking into account proper usage of computing performance of the hardware resources, memory, processor unit, graphics power and so forth. We execute the models on a PC station with 32bit Windows OS, 4G memory and double processor central unit. This hardware setup manages to handle dynamic instantiation of up to 10000 objects in a 3D grid space with 25 grid-pixel on each axis (total 15625). It further maintains several 3D matrices of the same size as the grid that are consists of integers for values of cell-extrinsic factors. We believe that a major limitation of the approach is computational power, since in the future we would like to greatly extend the number of entities in the models. We believe that distributed algorithms and memory management tools could help to confront these future challenge.

## 3. Model for Pancreatic Organogenesis

We have employed the generic reactive animation setup to model the development of the pancreas, a highly complex system containing numerous objects. The pancreas is an essential organ that is involved in regulation of metabolic and digestive pathways. During development, it takes on an interesting three-dimensional cauliflower-like shape. A prerecorded run of the simulation is available at `www.wisdom.weizmann.ac.il/~yaki/runs`

Abnormal functioning of the pancreas leads to lethal diseases such as pancreatitis and diabetes. Our model includes a comprehensive state-based specification that results from analyzed scientific data. The model is linked to an animated front-end and a mathematical analysis GUI. In the future, we plan to link these two also with a scenario-based specification to simulate mutations (e.g., defective blood vessels). In [33], we provide details of this work and discuss the intriguing ideas and predictions that emerged from the model. See `www.wisdom.weizmann.ac.il/~yaki/abstract/` for a short description.

**Model for pancreatic development:** Using the language of Statecharts, we modeled pancreatic development as an autonomous agent system [3] in which cells are autonomous entities that sense the environment and act accordingly. Our Statechart model defines the behavior of a cell element using a hierarchy of states with transitions, events, and conditions for biological stages and events. The statechart model for the cell object in this model consists of orthogonal components that designate behaviors for the various mechanisms of the cell (see schematic version Fig. 1). The stubbed arrow in each component of the statechart designates the initial state; that is, the state the cell enters when it is first executed.
We compiled the model using Rhapsody into executable reactive machine code (in our case in C++). Cells are considered to be the basic objects of the model, and the progress of the simulation/execution

relies very much on their behavior. An execution of the model is initiated with approximately 500 cells, which, with the aid of additional processes, proliferate and create new instances. A typical execution ends with approximately $10,000$ objects.

In the statechart model the behavior of a receptor, which is responsible for perceiving external signals, is defined as an independent component that can be either in state `Unbound` or in state `Bound`. Similarly, a gene defines an independent component that can be either in state `Expressed` or in state `Unexpressed`. Other components describe behaviors for various molecular mechanisms (such as, differentiation, proliferation, death) in a cell during its lifespan. The `Differentiation` component for example defines the stages in the development of the cells (e.g., `Endodermal`, `Pancreatic Progenitor` etc.). The `Proliferation` component define the different stages in the cell cycle, which a cell takes when dividing.

In the current stage the cell incorporates biological data that is relevant to the early stages of pancreatic organogenesis. However, we are currently extending the model to cover more information from later stages. A long run challenge would be to extend the design to cover more data or gene expression in the entire genome in the course of developing a generic stem cell.

The environment was modeled as a computational 3D grid that overlies the position of the cells. The grid contains data regarding the position of the cell and relevant tissues. Thus for example, the environment indicates the position of the notochord tissue in that grid, as well as that of other relevant tissues. The environment also stores in the grid concentrations for factors that direct the proliferation, differentiation and motility of cells. The model updates these concentrations in the ECM grid cubes, as the development the source tissue develops. For example, the notochord secretes several factors in the extracellular space. Thus, in our model, the notochord object regulates concentrations of relevant factors in the ECM grid next to its specified location.

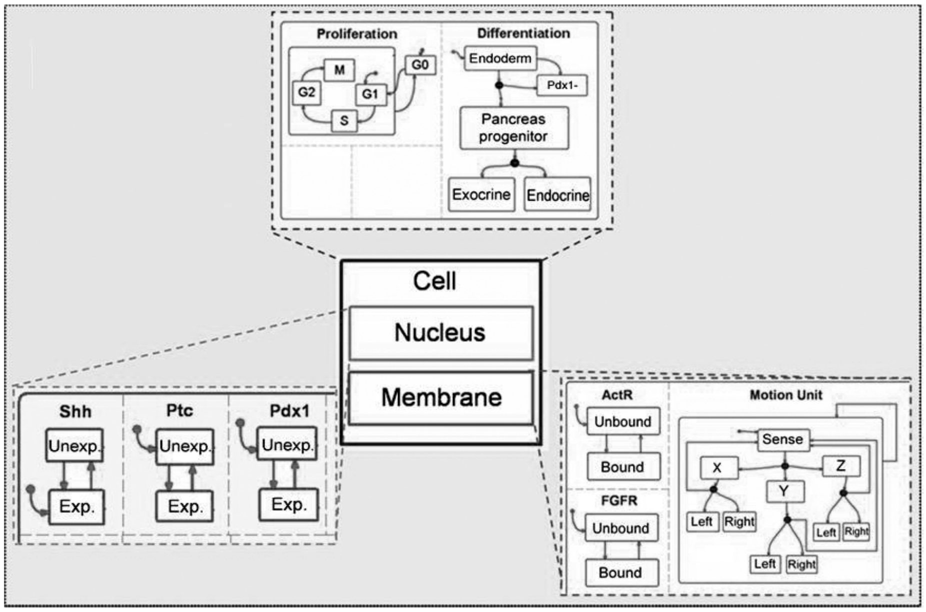

Figure 1. The model for a cell as an autonomous agent accompanied by its visualization (top-left)

**Designing an animated front-end:** We visualize the simulation in an animated front-end that we built based on what is depicted in the literature. Each of the participating components is represented as a 3D element possessing attributes to represent change in location and behavior (Fig. 2A). For example, the cells are represented in the front-end as spheres; at run-time, an instance of a cell directs its corresponding animated sphere according to its active state. A differentiated cell might, for example, change its color to depict the new stage. As the simulation advances, the cells dynamically act in concert to form the cauliflower-shape structure of the pancreas (Fig. 2B). At any stage, the user can halt the simulation and query objects, or interact directly with the emerging structure (e.g., 'slice' the structure) (Fig. 2C).

**Mathematical analysis user interface:** We designed a MATLAB GUI to provide mathematical analysis of the simulation. The GUI continuously receives data from the reactive engine, analyzes it and provides graphs and statistics (depicting, for example, cell population, proliferation rates, etc.). It thus enables the user to evaluate the dynamics of the simulation over time. Fig. 2D shows a snapshot of a graph that displays the cell count over time. In general, the system developer can design many graphs and statistics, which are related to the relevant system and whose data is gleaned from the model.

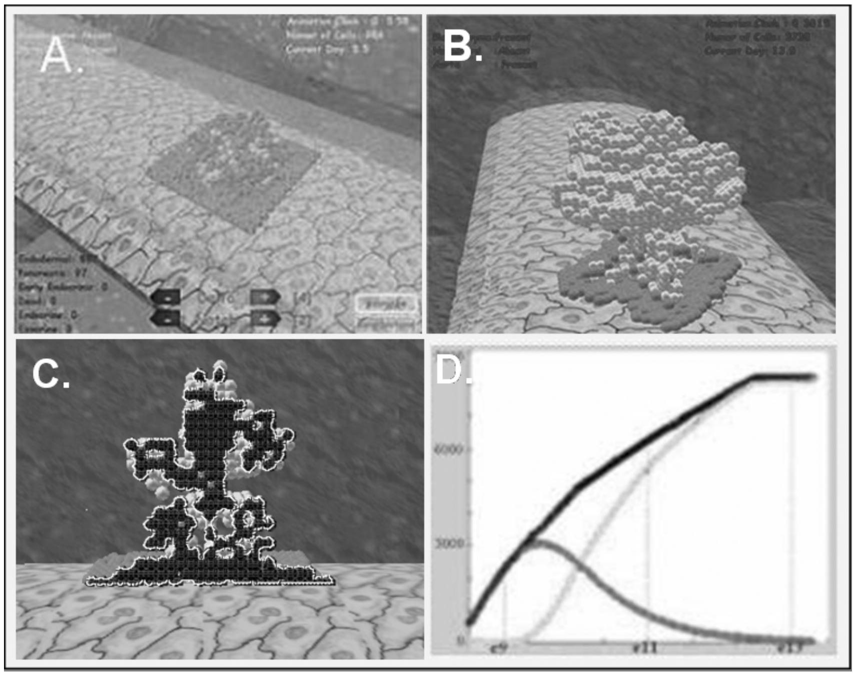

Figure 2. Animating the pancreas. A. 3D animated front-end for the pancreatic development. B. The pancreatic structure emerging from the simulation. C. User interaction with the simulation, the simulation was halted and a dynamic cross-section slicing was triggered. D. Mathematical analysis of pancreatic development: number of cells as function of time.

**The pancreas at run-time:** Once the model is executed, instances of the `Cell` are created and appear in the front-end as a sheet of red spheres at the proper location on the flat endodermal `Gut`. Once a `Cell` instance is created, one state in each concurrent component of its statechart is set to be the active state.

At this point, the `Cells` are uniform and their active states are set to the initial states. In parallel, the environment is initiated and defines the initial concentrations of factors in the extracellular space. As the simulation advances, cells respond to various events (e.g., the concentration of factors in their close vicinity) by changing their active states accordingly. Hence, the cell sheet loses uniformity at a very early stage of the simulation.

As the simulation advances, among other things, cells differentiate, proliferate and move in response to various signals. These processes are driven by many extracellular events (e.g., from the membrane) and intra-cellular events (e.g., from the nucleus). These events change the active states in orthogonal pieces of the specification through the different stages of the life cycle of a cell. For example, the proliferation process is initiated by extra-cellular signals when the membrane senses relevant factors and generates a chain of inter-cellular events (in the cell and the nucleus) that promote cell division. The process ends when the `Cell` duplicates itself by creating an identical instance. In turn, a message is sent to the front end, which creates a new identical sphere corresponding to the new `Cell` at the proper location. Similarly, when cell-extrinsic cues drive cell-intrinsic regulations that promote differentiation, the cell changes its differentiation stage (i.e., the active state in the `Differentiation` component) and the corresponding animated figure displays the change (e.g., by changing position, color, etc). A typical example is the specification process an endodermal cell goes through when it adopts a pancreatic progenitor fate, in which cues from the notochord promote the expression of the pancreatic marker, which sets the active developmental stage to pancreatic progenitor. The front-end, in turn, changes the color of the corresponding cell from red to green.

The cell population acts in concert to drive the simulation by promoting various decisions in individual cells. Consequently, messages are continuously sent between the different components in the architecture to drive the simulation. These processes are displayed in parallel in each one of the components. The state-based specification highlights the active state of the various objects, and at the same time the front-end and the mathematical GUI continuously visualize and analyze the simulation.

One challenge is to extend the model to support scenario-based specifications to drive mutated behaviors in the simulation. The user will be able to trigger mutation scenarios through the various tools in the model, and to observe the mutated behavior through the animation and analysis. Such *in silico* experiments may be tested in-vivo on the real system through laboratory collaboration.

**Studying the model:** To test the simulations, among other things, we compared the emerging structure against histological sections of the pancreas at different stages. One *in silico* experiment is shown in Figure 3. This test compares the simulation against a 2D histological image of pancreatic morphogenesis at the tenth embryonic day. The histological cross-section of a pancreatic bud is given in Figure 3, top-left. Cells that express the key pancreatic gene Pdx1 are stained green, whereas, other cells are shown in red. This image is compared against two different perspectives of the simulation in Figure 3, bottom and a cross-section image of the structure in Figure 3 right-top. The simulation corresponds well with the histology, indicating that the 3D structure emerging from the simulation indeed captures pancreatic morphogenesis in the mouse.

During the morphogenetic analysis, we noticed that the simulation, although never explicitly programmed to do so, exhibits clusters of pancreatic cells that do not express the pancreatic marker (Figure 3, bottom). These clusters are reminiscent of a group of cells observed *in vivo* in the early pancreas, termed *primary transition cells*. The *in silico* clusters aggregated at the top of the bud similar to clusters of primary transition cells in vivo. Using our model, we studied the appearance of these clusters and predicted that the origin of this population is from cells that never expressed the pancreatic marker. The model also shows that these clusters are formed by cells with a common ancestor and are thus a consequence of proliferation rather than aggregation. We are currently collaborating with the group of Dr. Yuval Dor at the Hebrew University to develop ways to test this prediction experimentally.

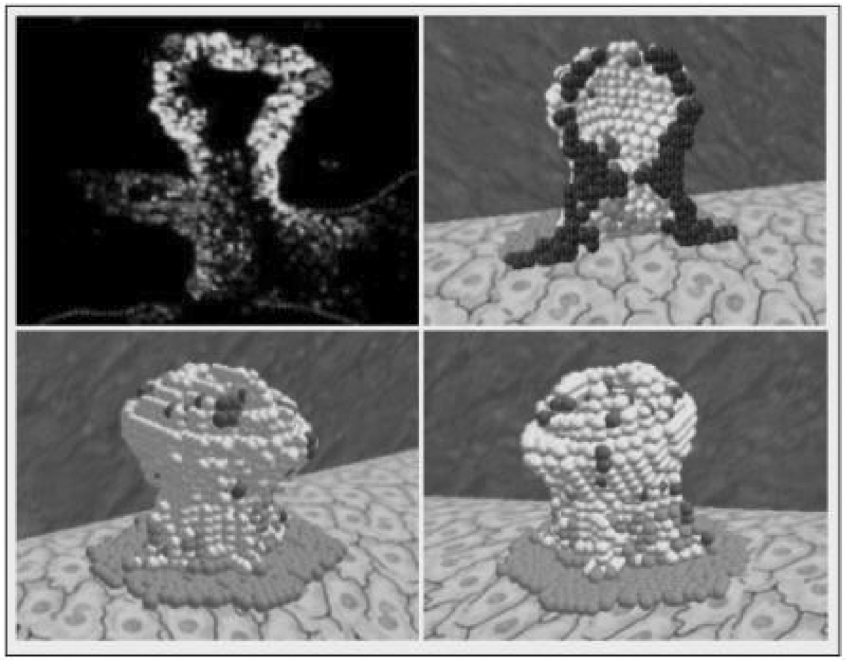

Figure 3. Histological cross-section image vs. the simulation at embryonic day 10. Notice the emerging Pdx1-negative red clusters in the simulation.

## 4. Animating GemCell

GemCell [1] is an ongoing project carried out in our group. Its goal is the development of a generic modeling and simulation tool (called GemCell, for Generic Executable Modeling of Cells). GemCell consists of a generic statechart model of a functioning cell that 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 information about the specific cellular system being modeled. Modeling a particular segment of biology involves setting up the DBS to contain data about the specific behaviors and responses of the particular kinds of cells in the system being modeled, and GemCell takes care of the integration at run-time with the statechart model. (See [1] for more detailed information.)

Recently, we employed the generic reactive animation platform to link GemCell to a 2D animated front-end (in Flash). The front-end animates simulations in the GemCell platform by visualizing various behavioral aspects of cells. For example, a cell is designated as a circle that changes its color to indicate differentiation. At run-time, the animation shows how the behavior of cell population emerges from the behavior of individual cells. Figure 4 provides snapshots of four time-steps of a representative execution of GemCell. This work will be described in more detail separately.

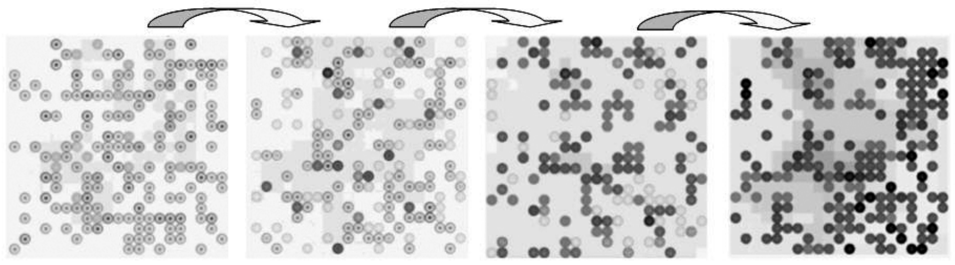
Figure 4. An example of a possible output of the GemCell simulation: snapshots at four time-steps over time.

## 5. Discussion

The last decade or so has seen increased interdisciplinary work on modeling various aspects of biological systems. These models are aimed at providing explanations for relevant experimental findings (see e.g., [2, 5, 15, 25]). Much of this work describes the system at a specific scale and often focuses on a single mechanism, ignoring multiple and multi-level interactions. Another type of modeling work formalizes gene expression and protein activity using a variety of mathematical and computational tools (for example, see [4, 23, 28, 29, 30]). An impressive example of an attempt at comprehensive modeling of an entire organ is the beating heart project [27], which has resulted in a formalization of the electric activities in the heart. However, by its mathematical nature, the model is not interactive and does not support the kind of interactive and manipulatable visualization we seek.

Recently, various papers describe computational modeling approaches for biological systems. In [14], hybrid automata are used to model the Delta-Notch mechanism, which directs differentiation in various biological systems. In [12], computational challenges of systems biology are described and various approaches for achieving them are discussed. Similar motivation for model-driven engineering approaches is discussed in [31]. In [38], a model for a eukaryotic cell is built, in which a UML class diagram was used to formalize the relations between a cell and its sub-cellular elements. The setup was empowered by specifying behavior of different cell types (e.g., red blood cell) using the ROOM formalism. A similar approach was employed in [36] to model the Ethylene-Pathway in Arabidopsis thaliana using statecharts and Live Sequence Charts.

Here, we describe the way we employ the reactive animation idea [11] and its generic architecture [22] to model biological systems. The pancreas model shows how a reactive engine controls the simulation while an animated front-end monitors the visualization, and makes it possible to model large-scale biological systems with many objects and interactions. We further describe how the GemCell project uses the same implementation to simulate cell behavior.

Among other things, using reactive animation for computational modeling of biological systems provides a visualized execution of numerous basic elements. Often, the animated running simulations disclose properties that have not been explicitly programmed into the model. Rather, they emerge from the concurrent execution of cells as a population. For example, in the pancreas model we found that concurrent execution of pancreatic cells gives rise to a property that corresponds well with first transition clusters found to appear early in the developing organ *in vivo* [33]. In general, since emergent properties are dynamic facets of a population, it is difficult to predict them from the model's static specifications. The animated front-end, which visualizes the simulation, can often disclose such phenomena. Biological explanations can then be traced by a precise examination of the relevant literature.

Computational modeling using reactive animation faces further challenges and extensions. One possible extension could be specifying scenarios in the Live Sequence Charts and execute them in the Play-Engine reactive engine to specify changes in the system [20]. Another challenge would be embedding quantitative aspects into the reactive models. One approach to achieve goal is by formalizing simple mathematical models (e.g., using differential equations) that are to be solved using parameters provided by the simulation at run-time. For example, once a cell expresses a gene, the model assigns the relevant parameters in a differential equation for gene expression. We believe that in the course of computational modeling of additional biological systems, the issue of quantitative analysis will arise in full force and various solutions will be tested and used.

We plan to continue this work in two separate directions. First, we are in the process of extending the coverage of the aforementioned models. We believe that in the long run the pancreatic model and the animated GemCell may serve for carrying out *in silico* experiments that may lead to a better understanding of the relevant biological systems. Second, generic reactive animation may serve as a starting point for modeling other biological systems, and it may help in the efforts to design an *in silico* organ or organism (see e.g., [26, 18, 7, 8, 17, 6, 39]). To this end, we are in the process of examining the possibility of using the architecture to model various aspects in the development of the *C. elegans* nematode, which is extensively used as a model organism in molecular and developmental biology.

## 6. Acknowledgments

Special thanks go to Hila Amir-Kroll and Avital Sadot, who are the key figures in the GemCell project. Thanks also to Shahar Maoz for comments on this manuscript.

## References

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

[2] Axelrod, J. D.: Cell Shape in Proliferating Epithelia: A Multifaceted Problem, *Cell*, **126**, 2006, 643–645.

[3] Brooks, R. A.: Elephants Don't Play Chess, *Robotics and Autonomous Systems*, **6**, 1990, 3–15.

[4] Cardelli, L.: Abstract Machines of Systems Biology, *T. Comp. Sys. Biology*, **3**, 2005, 145–168.

[5] Ciliberto, A., Novak, B., Tyson, J. J.: Mathematical Model of the Morphogenesis Checkpoint in Budding Yeast, *J Cell Biol*, **163**, 2003, 1243–1254.

[6] Cohen, I. R.: Tending Adam's Garden: Evolving the Cognitive Immune Self, *London: Academic Press*, 2000.

[7] Cohen, I. R.: Modeling immune behavior for experimentalists, *Immunol Rev*, **216**, 2007, 232–236.

[8] Cohen, I. R., Harel, D.: Explaining a Complex Living System: Dynamics, Multi-scaling and Emergence, *J R Soc Interface*, **4**, 2007, 175–182.

[9] Edelman, L. B., Chandrasekaran, S., Price, N. D.: Systems biology of embryogenesis, *Reprod Fertil Dev*, **22**(1), 2010, 98–105.

[10] Efroni, S., Harel, D., Cohen, I. R.: Toward Rigorous Comprehension of Biological Complexity: Modeling, Execution, and Visualization of Thymic T-Cell Maturation, *Genome Res*, **13**, 2003, 2485–2497.

[11] Efroni, S., Harel, D., Cohen, I. R.: Reactive Animation: Realistic Modeling of Complex Dynamic Systems, *IEEE Computer*, **38**, 2005, 38–47.

[12] Finkelstein, A., Hetherington, J., Li, L., Margoninski, O., Saffrey, P., Seymour, R., Warner, A.: Computational Challenges of Systems Biology, *IEEE Computer*, **37**(5), 2004, 26–33, ISSN 0018-9162.

[13] Fisher, J., Henzinger, T. A.: Executable cell biology, *Nat Biotechnol*, **25**(11), 2007, 1239–1249.

[14] Ghosh, R., Tomlin, C.: Symbolic Reachable Set Computation of Piecewise Affine Hybrid Automata and its Application to Biological Modelling: Delta-Notch Protein Signalling, *Syst Biol (Stevenage)*, **1**, 2004, 170–183.

[15] Gibson, M. C., Patel, A. B., Nagpal, R., Perrimon, N.: The emergence of geometric order in proliferating metazoan epithelia, *Nature*, **442**, 2006, 1038–1041.

[16] Harel, D.: Dynamic Logic, in: *Handbook of Philosophical Logic Vol. II* (D. Gabbay, F. Guenthner, Eds.), Reidel, 1984, 497–604.

[17] Harel, D.: A Grand Challenge for Computing: Full Reactive Modeling of a Multi-Cellular Animal, *Bulletin of the EATCS*, **81**, 2003, 226–235.

[18] Harel, D.: A Turing-like test for biological modeling, *Nat Biotechnol*, **23**, 2005, 495–496.

[19] Harel, D., Gery, E.: Executable Object Modeling with Statecharts, *Computer*, **30**, 1997, 31–42, Also in *Proc. 18th Int. Conf. Soft. Eng.*, Berlin, IEEE Press, March, 1996, pp. 246–257.

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

[21] Harel, D., Pnueli, A.: On the Development of Reactive Systems, *Logics and Models of Concurrent Systems* (K. R. Apt, Ed.), F-13, Springer-Verlag, New York, 1985.

[22] Harel, D., Setty, Y.: Generic Reactive Animation: Realistic Modeling of Complex Natural Systems, *Formal Methods in Systems Biology*, 5054, 2008.

[23] Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic Model Checking of Complex Biological Pathways, *Proc. Computational Methods in Systems Biology (CMSB'06)* (C. Priami, Ed.), 4210, Springer Verlag, 2006.

[24] Kam, N., Kugler, H., Marelly, R., Appleby, L., Fisher, J., Pnueli, A., Harel, D., Stern, M. J., Hubbard, E. J.: A scenario-based approach to modeling development: a prototype model of C. elegans vulval fate specification, *Dev Biol*, **323**(1), 2008, 1–5.

[25] Nelson, C. M., Vanduijn, M. M., Inman, J. L., Fletcher, D. A., Bissell, M. J.: Tissue geometry determines sites of mammary branching morphogenesis in organotypic cultures, *Science*, **314**, 2006, 298–300.

[26] Noble, D.: Modeling the Heart-from Genes to Cells to the Whole Organ, *Science*, **295**, 2002, 1678–1682.

[27] Noble, D.: The heart is already working, *Biochem Soc Trans*, **33**, 2005, 539–542.

[28] Priami, C., Quaglia, P.: Modelling the dynamics of biosystems, *Briefings in Bioinformatics*, **5**, 2004, 259–269.

[29] Regev, A., Shapiro, E.: Cellular abstractions: Cells as computation, *Nature*, **419**, 2002, 343.

[30] Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the pi-calculus process algebra, *Pacific Symposium on Biocomputing*, 2001.

[31] Roux-Rouquié, M., da Rosa, D. S.: Ten Top Reasons for Systems Biology to Get into Model-Driven Engineering, *GaMMa '06: Proc. of the 2006 international workshop on Global integrated model management*, ACM Press, New York, NY, USA, 2006, ISBN 1-59593-410-3.

[32] Sadot, A., Fisher, J., Barak, D., Admanit, Y., Stern, M. J., Hubbard, E. J. A., Harel, D.: Toward Verified Biological Models, *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, **5**(2), 2008, 223–234, ISSN 1545-5963.

[33] Setty, Y., Cohen, I. R., Dor, Y., Harel, D.: Four-dimensional realistic modeling of pancreatic organogenesis, *Proc Natl Acad Sci U S A*, **105**(51), 2008, 20374–20379.

[34] Swerdlin, N., Cohen, I. R., Harel, D.: The Lymph Node B Cell Immune Response: Dynamic Analysis in-silico, *Proceedings of the IEEE, special issue on Computational System Biology*, 96, 2008.

[35] Täubner, C., Eckstein, S.: Signal Transduction Pathways as Concurrent Reactive Systems: A Modeling and Simulation Approach Using LSCs and the Play-Engine, *Electr. Notes Theor. Comput. Sci.*, **194**(3), 2008, 149–164.

[36] Täubner, C., Merker, T.: Discrete Modelling of the Ethylene-Pathway, *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, IEEE Computer Society, Washington, DC, USA, 2005, ISBN 0-7695-2657-8.

[37] Telelogic, www.telelogic.com.

[38] Webb, K., White, T.: Cell Modeling with Reusable Agent-based Formalisms, *Applied Intelligence*, **24**(2), 2006, 169–181, ISSN 0924-669X.

[39] Zhang, X., Dragffy, G., Pipe, A. G.: Embryonics: A Path to Artificial Life?, *Artif Life*, **12**, 2006, 313–332.