Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 194 (2008) 119–131
www.elsevier.com/locate/entcs

Concurrency in Biological Modeling: Behavior, Execution and Visualization

David Harel[a,1,2] Yaki Setty[a,3] Sol Efroni[b,4]
Naamah Swerdlin[a,5] and Irun R. Cohen[c,6]

[a] *Department of Computer Science and Applied Mathematics*
Weizmann Institute of Science
Rehovot, Israel

[b] *National Cancer Institute Center for Bioinformatics*
Bethesda, MD, USA

[c] *Department of Immunology*
Weizmann Institute of Science
Rehovot, Israel

Abstract

Modeling natural systems is a complicated task that involves the concurrent behavior of various processes, mechanisms and objects. Here, we describe an approach that we have been taking in our group for several years, whereby the complexity of the problem is reduced by decomposing a natural system into its basic elements, which are then reassembled and combined to form a comprehensive, simulatable model of the system. Our modeling approach allows one to view a natural system at various levels of abstraction, in a way that makes it possible to zoom in and out between levels. Using statecharts, a high level visual formalism, we specify the behavior of the basic elements of each level and compile these into executable code, which is then linked to an animated front-end. At run-time, the concurrent execution of the basic elements is continuously displayed and provides a dynamic description of the system. We illustrate this approach by modeling aspects of three biological systems: development of the mammalian pancreas; the differentiation of T cells in the thymus; and the dynamic architecture of a lymph node. We compared each model's behavior with experimental data and also reproduced genetic experiments *in silico*. Interestingly, certain behavioral properties that were not explicitly programmed into the model emerge from concurrent execution and correspond well with the experimental observations.

Keywords: Reactive Animation, reactive systems, biological modeling, statecharts, concurrent modeling

[1] Part of this author'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.

[2] Email: david.harel@weizmann.ac.il

[3] Email: yaki.setty@weizmann.ac.il

[4] Email: sefroni@mail.nih.gov

[5] Email: naamah.swerdlin@weizmann.ac.il

[6] Email: irun.cohen@weizmann.ac.il

[7] 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.

doi:10.1016/j.entcs.2007.12.009

1 Introduction

An ambitious and long-term goal is to comprehensively model an entire organism, which can be viewed as a very complex system with many interacting concurrent objects [13]. We describe here our approach to such modeling, developed in our group for the past several years. As a first step towards an *in-silico* organism, we have reduced the complexity of the problem by modeling particular organs of the complete organism [4]. We then can combine the various organs concurrently to discover how they act in concert to form and drive an organism. Note that any organ is itself a compound element, with many internal concurrent processes and mechanisms. We reduce the organ's complexity by modeling its basic concurrent elements, molecules and cells. Again, however, each cell has many concurrent subcellular elements (e.g., the nucleus) that drive its development and its life history. Figure 1 illustrates these levels of abstraction in natural systems. The levels exist concurrently, of course, and, in addition, each level consists of many concurrent manifestations of its basic elements. We deliberately leave open both ends of the

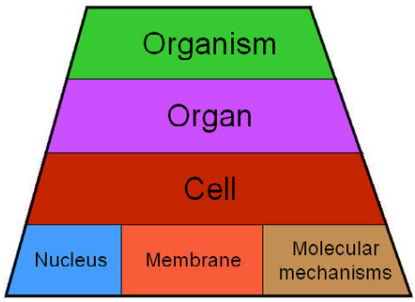

Fig. 1. Various levels of abstraction in biological systems.

pyramid to indicate that the pyramid is incomplete and can be extended in both directions. Downwards, we may zoom in and model gene expression and protein activity using a variety of mathematical and computational tools (for example, see [3,19,25,26,27]). Such extensions address the task of modeling the behavior of many concurrent molecules as they are expressed in the cell and interact over time. Conversely, we may zoom out and add upper levels to the pyramid. Thus, we can model concurrent interactions between different organisms, not necessary of the same species, in a population. A comprehensive model of a mouse, for example, may be used to specify the interactions within a population of mice and may help in understanding how the population behaves under various circumstances, such as starvation.

In this paper, we focus on two levels of abstraction — the cell and the organ — and describe an approach to specify their concurrent behavior using high-level visual formalisms. The specifications are compiled into executable machine code and enable concurrent execution of objects. Moreover, the generated program is linked to an animated front-end that visualizes the behavior of the concurrent execution in real-time. We illustrate this approach by discussing briefly models of three different biological systems, with different characteristics and behaviors, that were developed

in our group over the last few years. One of our models, pancreatic organogenesis [30], captures cells cooperating dynamically to form an organ. The other two, maturation of T-cells in the thymus gland[6] and development of the lymph node[31], simulate how cells behave in a mature organ. The models were tested by comparing computer executions with previous investigations, including histological images, experimental data, and theoretical models. Furthermore, we tested the models using *in silico* knock-out experiments in which we disabled elements in the model and studied the effect on the system. Interestingly, some unexpected behavioral properties emerged from the simulations at run-time. These emergent properties were not explicitly programmed by us, but are a consequence of the concurrent execution of basic elements with identical specification. Here, we focus on emergent properties at the level of the organ, however the emergence is identified across different levels of abstraction, as discussed in [4].

2 Tools and Methods

To model the concurrent mechanisms within and between different levels of abstractions of natural systems, we use the visual formalism of *statecharts* [12,16] as it is implemented in the Rhapsody tool [33]. Statecharts define behavior using a hierarchy of states with transitions, events, and conditions, and, using Rhapsody (or other similar tools) Statecharts can be compiled into executable reactive machine code (for example, in C++).

By its nature, the language of Statecharts enables the specification of orthogonal state components within a statechart. Each component specifies one parallel aspect of the behavior of the object. At run-time, one state in each component is defined as an active state – that is the current state of this component. Thus, the state of an object is identified by the set of active states of its components. As the simulation advances, various events move the active state in each orthogonal component from one state to another. Thus, concurrent execution is naturally implemented in the executable code that is generated from the specification.

To visualize the model, we use the idea of *reactive animation* (RA) [7,15], a technique that links a reactive executable model with an animated front-end to form a visualized, interactive, and dynamic 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, we have upgraded the implementation resulting in a generic platform that enables interaction between various tools such as multiple reactive engines, 3D animation, real-time analysis and more [18].

3 Modeling Concurrency: The Basic Approach

The organ and the cell levels of abstraction involve many concurrent processes and objects. In this section, we discuss the concurrent behavior of the basic elements in each level and the way concurrence is specified in our models. Furthermore, we

illustrate the concurrent execution of specifications across the two levels by a small representative example of a conceptual biological process.

3.1 Concurrent behavior in a eukaryotic cell

A cell consists of many concurrent sub-cellular and molecular mechanisms that drive development and function over its lifespan. Each sub-cellular element consists of many concurrent processes and mechanisms that dynamically drive the cell's function over time. We specified behavior for two sub-cellular elements of a cell, namely the nucleus and membrane. The `Cell` object itself specifies the behavior of molecular mechanisms in the cell (e.g., proliferation). This setup formalizes a cell as an autonomous agent[2] that senses its environment and acts based on its specification. This setup is illustrated in Figure 2, which shows the elements accompanied by schematic versions of their statecharts. Each orthogonal component in the statechart of the `Cell` specifies the behavior of a concurrent molecular mechanism. Similarly, in the statecharts of the `Nucleus` and the `Membrane`, each orthogonal component specifies concurrent behavior of a `Gene` and `Receptor`, respectively. The front-end visualizes the `Cell` and holds relevant structural information. As the simulation progresses, the animated `Cell` changes its properties (e.g., color) to indicate behavioral changes in the object.

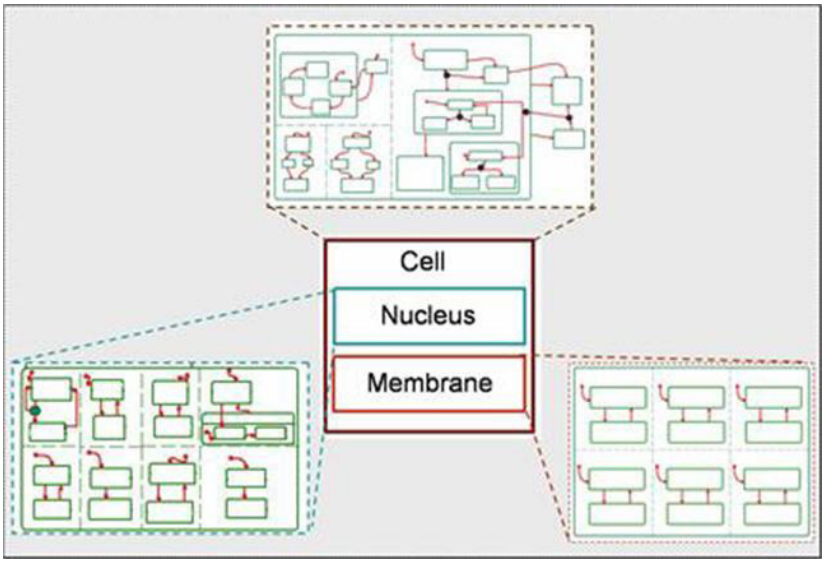

Fig. 2. An autonomous cell (from [30]): Three concurrent elements accompanied by a schematic statechart for their behavior: the Nucleus, the Membrane and the element, which formalizes behavior of different molecular mechanisms (e.g., differentiation). Notice the dashed line that indicates orthogonality; i.e., these state components are independent and act concurrently.

The membrane, the surface that acts as the boundary of a cell, contains many concurrent receptors, which are responsible for perceiving external signals. The receptors are located on the membrane and are continuously searching for molecules in the surrounding environment. Each receptor recognizes specific molecules that may bind to it, and the binding activates signaling pathways that regulate molecular

mechanisms in the cell. To model the membrane, we defined each receptor as an independent component that can be in the `Unbound` or in `Bound` state, with the difference being reflected in the dedicated element on the cell surface in the front-end; see Figure 3,top.

Similarly, the nucleus, the core of a cell that contains the DNA, consists of concurrent genes that regulate its development. Genes are expressed in response to various signals in the cell. Genes express proteins that influence the cell's behavior. To model the nucleus, we took a simplistic approach, defining each gene as an independent component that can be either `Expressed` or `Unexpressed`. The effect of gene expression is diverse and depends on the process, and is visualized in different manners. For example, in the case of markers, proteins that determine differentiation, expression is visualized by color changes of the animated cell; see Figure 3,middle.

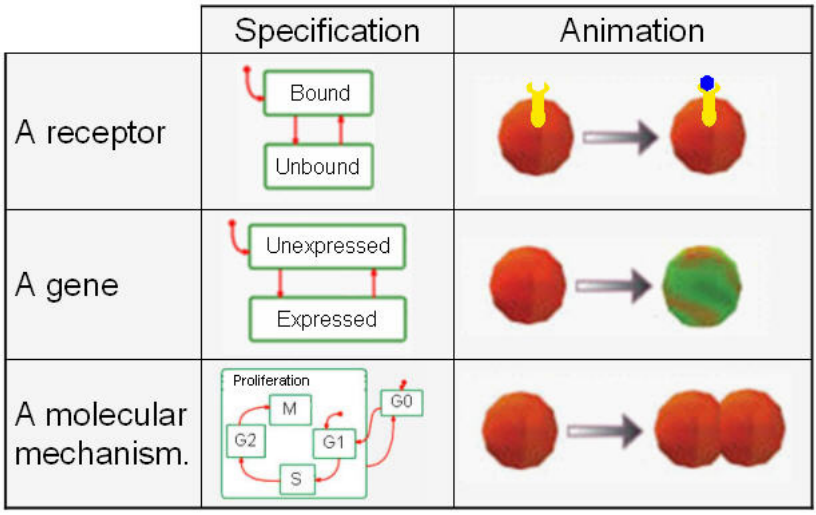

Fig. 3. Basic components of a `Cell`: a receptor, a gene and the cell cycle mechanism.

The `Cell` itself describes the behavior of various molecular mechanisms (such as, differentiation, proliferation, death) in a cell during its lifespan. This element also carries the spatial 3D coordinates of the cell and updates their values at run-time as the simulation progresses. We specify the mechanisms as orthogonal components, which at run-time act concurrently to drive the cell's behavior over time. As an example, consider cell proliferation, illustrated in Figure 3, bottom. The `Proliferation` component defines a state for each stage of the cell cycle. At run-time, at the end of the `Proliferation` stage (when its active state moves to state `M`)), the `Cell` duplicates itself by creating an identical `Cell` instance (i.e., a new instance of a cell is created and its active states are set to be identical to its parent). At the front-end, an additional animated cell is created in the appropriate location, which was calculated during its parent division. Furthermore, at run-time, when the `Membrane` senses changes in specific environmental conditions, an event is generated and blocks expression specific `Genes`, which are crucial to proliferation. In turn, the period of specific proliferation stages is extended and the process is

decelerated. Similarly, expression of other `Genes` triggers an event, which decreases the proliferation period and leads to acceleration of the process. Other mechanisms, such as signaling pathways or differentiation are specified in a similar way.

3.2 Concurrent behavior in an organ

The function and development of an organ are largely dependant on interactions between cells and their environment. Very often the environment consists of several tissues that generate various signals (e.g., secreted molecules) and thus affect nearby cells and tissues. The various objects of the organ, mainly cells, receive the signals and cooperate to form the organ and drive its behavior over time.

Inspired by ideas from the Turing instability hypothesis [34,35], we formalize surrounding environments as a 3D grid that overlies the organ. Accordingly, the space surrounding the organ is divided into 3D grid-cubes with a fixed volume. Objects, representing different elements in the environment, regulate concentrations of factors that are stored in grid-cubes. At run-time, autonomous cells sense the various factors and act accordingly. The animated front-end can then visualize each element of the model and expose, at run-time, the formation and behavior of the organ. The visualization is best set up to be consistent with illustrations and descriptions of the system as depicted in the biological literature.

Figure 4 illustrates, as an example, the setup of the organ level in our pancreatic organogenesis model [30]. The model consists of three different tissues, the `Aorta`, the `Notochord` and the `Mesenchyme`, which regulate the development of the pancreas; see Figure 4, top. Each tissue is represented as a concurrent object that can regulate factors in the grid, which is stored in the `Extracellular Matrix` object; Figure 4, middle. Thus, as in nature, tissues and cells interact through the extracellular matrix, rather than directly. The animated front-end (Figure 4, bottom) visualizes the extracellular space by representing each element as an animated figure that changes its properties over time. For example, the `Mesenchyme` is represented as a tissue-like space that changes its color when the `Aorta` is present.

3.3 Concurrent execution

When the model is executed, the environment is initiated and instances of the `Cell` are created and appear in the front-end at their initial positions. Once a `Cell` instance is created, one state in each concurrent component of its statechart is set to be an active state. At this point, the `Cells` are uniform and their active states are set to their initial states (designated in the statecharts by stubbed arrows). As the simulation advances, the cells respond concurrently to various events by changing their active states accordingly. Hence, the population loses uniformity at a very early stage of the simulation.

To illustrate the simulation in progress, consider a conceptual process that unifies many biological processes such as signaling pathways or cell migration. Such a process is stimulated by an external signal that initiates a chain of various interactions across the different levels of abstraction. At the cell level, concurrent

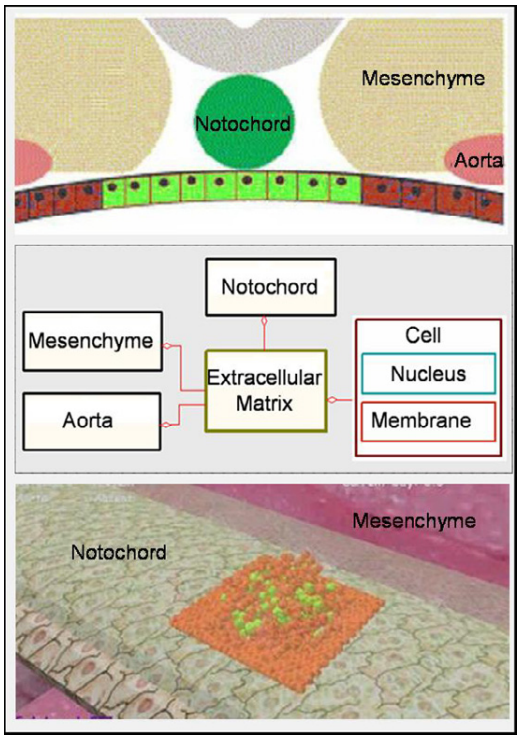

Fig. 4. Modeling the organ level in pancreatic organogenesis (from [30]). Top: An illustration of the participating tissues (adapted from [22]). Middle: The interaction scheme of the model. Bottom: The 3D animated front-end of the model.

independent `Receptors` on the `Membrane` sense stimulation signals in their vicinity (i.e., neighboring grid cubes) and change their active state to `Bound` accordingly. At the front-end, the corresponding `Receptors` reflect the changes by updating their relevant property. Consequently, events are sent to the `Nucleus`, which initiates expression in various concurrent `Genes`. In turn, the active state in the relevant components moves to the `Expressed` state and the corresponding animated cell changes its color. Eventually, an event is generated and the relevant molecular mechanisms move to new states. For example, an event may promote a cell to proliferate. Accordingly, the active state in the `Proliferation` component in the `Cell` becomes `M`, the cell duplicates itself, and a new animated cell appears in the front-end.

Zooming out reveals the development of the organ, in which interactions between concurrently active cells and various concurrent environmental elements drive the simulation. The concurrently active cells, as a population, act in concert until the simulated system achieves equilibrium. The front-end visualizes the process and displays the organ at work. Furthermore, one can analyze a model by reproducing genetic ablation experiments, in which an organism is engineered to lack one of its genetic components. Accordingly, we can disable the corresponding element and examine the effect of the concurrent execution under the mutated condition. Interestingly, concurrent execution of the basic elements often reveals properties that were not explicitly programmed into the model. Rather, they emerge from the

concurrent execution of cells as a population. In general, since emergent properties are dynamic properties of a population, it is rather difficult to predict them from the model's static specifications. At the animated front-end, which visualizes the simulation, the phenomenon is often easily seen and can then be carefully examined against the literature for a biological explanation.

4 Modeling Concurrency: Examples

We have illustrated our approach using three models developed in the last few years. We briefly describe the models, not in their chronological order. The first is a model of pancreatic organogenesis in the embryonic mouse [30], in which the concurrent execution of pancreatic cells forms the unique 3D structure of the organ (Figure 5A). The other two models relate to the immune system: one simulates the differentiation of T-cells in the thymus gland[6] and the other simulates the development and function of cells in the lymph node[31]. In both immune models, the concurrent execution describes the maturation of precursor cells in an active organ (Figure 5B and 5C, respectively).

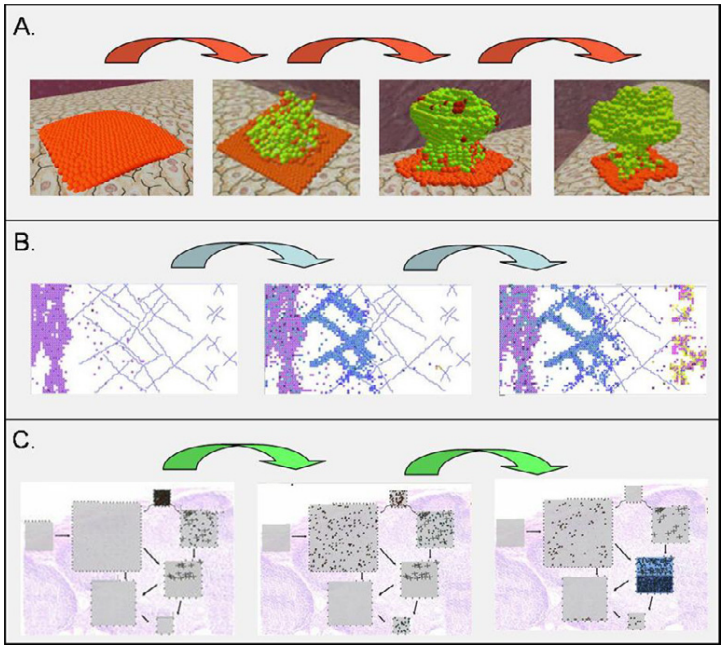

Fig. 5. Simulation of organ level in three models: (A) pancreatic organogenesis; (B) maturation of T-cells in the thymus; (C) development of the lymph node.

We studied the three models by comparing simulations against relevant experimental data and reproduced some genetic ablation experiments *in silico*. As an example, Figure 6A 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. Figure 6B shows a similar analysis of the

thymus model; microscopy image of the wild type thymus (left) compared with the same area in the model (middle). On the right, the figure 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.

Analysis of the models also revealed some very interesting emergent properties, which correspond well with biological phenomena. For example, in the pancreas model, we found that concurrent execution of pancreatic cells gives rise to a property that corresponds well with endocrinic clusters found to appear early in the developing organ *in vivo* [30]. Similarly, the concurrent execution of T-cell development in the thymus led to the emergence of competitive behavior between the cells [6]. We analyzed and studied these properties and suggested some insights into the phenomena [8,30].

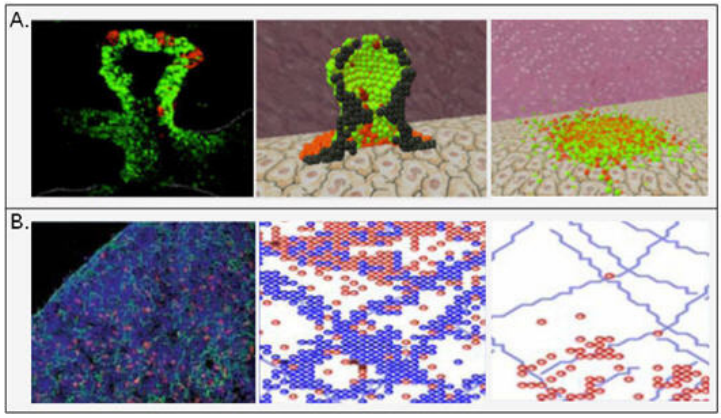

Fig. 6. Comparison between *in silico* and *in vivo* results. (A) Histological section of the developing pancreas (left; adopted from [20]), the emerging structure of the simulated pancreas (middle), and *in silico* genetic ablation of the `Aorta` (right). Notice the different background color indicating that the aorta is absent. (B) Microscopic image of the T-cells in thymus (left; adopted from [24]), a snapshot of the simulation (middle) and *in silico* knockout of the CXCR4 gene (right).

Recent work in our group, that was also inspired by the aforementioned approach, has produced a system we call *GemCell* [1]. 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. 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 under description. During execution, statecharts read in the specific data and the combination runs just as described in the particular models above. This project is still in its early stages of development.

5 Discussion

The recent decade has seen an increasing effort in modeling natural systems. Our approach is guided by the observation that natural systems can be beneficially

specified as reactive systems [13,14]. This observation has led to quite a lot of work on modeling biology using various software engineering tools and ideas to simulate concurrent behavior in various natural systems. Furthermore, this approach incorporates cental modeling elements, such as time and stochasticity (e.g., the simulation in different runs with same initial conditions does not necessarily give identical results).

In [9], the authors describe computational challenges of systems biology and discuss various approaches for achieving them. Further motivation for such a computational approach for modeling natural systems is given in [28]. A detailed distinction between the computational and mathematical approaches is given in [10], where the term *executable biology* is used to describe kinds of modeling carried out in our group, and recently also elsewhere.

[38] provides a model for eukaryotic cell in which 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. Similar approach was employed in [32] to model the Ethylene-Pathway in Arabidopsis thaliana using statecharts and LSCs. Another approach for modeling is described in [11], where hybrid automata were employed to model the Delta-Notch mechanism, which directs differentiation in various natural systems. This model reproduces the Delta-Notch decision: cells that express Delta are surrounded by cells that express Notch.

This paper emphasizes the idea of decomposing a natural system into its concurrent components, which are then modeled and reassembled to form comprehensive emerging simulations of the system. We recognize concurrency along two dimensions: *vertical concurrency* between different levels of abstraction, and *horizontal concurrency* between the basic elements on the sub-levels. Using this approach, we can specify behavior for concurrent elements and reassemble them horizontally and vertically to form a comprehensive model. Using reactive engines, we can then analyze and execute the specifications and observe the effect of the concurrent execution of the basic elements as a population.

The approach supports integration with other models that have designed using different approaches. Thus, for example, we may employ this approach for interfacing with the ROOM-based model of a cell in [36,37,38] or the hybrid automata based model for the Delta-Notch mechanism [11]. Also, within our own group, besides the examples discussed above that use statecharts, we have used the language of *live sequence charts* (LSCs) [5,17] to specify biological systems in an inter-object (rather than an intra-object) way (see [21]). These kinds of models can also be linked with stratified statechart-based modeling, as we have shown, for example, in [29].

As the field of computational modeling advances and more biological systems are modeled, we believe that these and other approaches to model natural systems (see, for example, [23]), may serve as basic elements in the effort to model a complete organism (see figure 7).

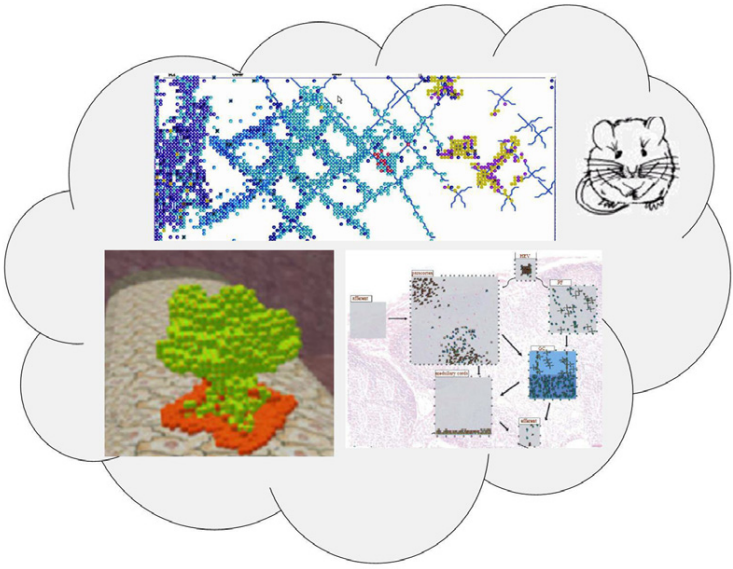

Fig. 7. Towards an *in silico* organism: three reactive animation models of different organs in a mouse. Top: T-cell maturation in the thymus. Bottom left: Pancreatic organogenesis. Bottom right: Development of a lymph node.

Acknowledgement

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

References

[1] Amir-Kroll, H., A. Sadot, I. Cohen and D. Harel, *GemCell: A Generic Platform for Modeling Multi-Cellular Biological Systems*, T. Comp. Sys. Biology **to appear** (2007).

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

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

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

[5] Damm, W. and D. Harel, *LSC's: Breathing Life into Message Sequence Charts*, in: P. Ciancarini, A. Fantechi and R. Gorrieri, editors, *Proc. 3rd IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'99)* (1999).

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

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

[8] Efroni, S., D. Harel and I. R. Cohen, *Emergent Dynamics of Thymocyte Development and Lineage Determination*, PLoS Comput. Biol. **3** (2007), pp. 127–136.

[9] Finkelstein, A., J. Hetherington, L. Li, O. Margoninski, P. Saffrey, R. Seymour and A. Warner, *Computational Challenges of Systems Biology*, IEEE Computer **37** (2004), pp. 26–33.

[10] Fisher, J. and T. A. Henzinger, *Executable Biology*, in: *WSC '06: Proc. of the 38th conference on Winter simulation* (2006), pp. 1675–1682.

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

[12] Harel, D., *Statecharts: A Visual Formalism for Complex Systems*, Sci. Comput. Programming **8** (1987), pp. 231–274.

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

[14] Harel, D. and A.Pnueli, *On the Development of Reactive Systems*, in: K. R. Apt, editor, *Logics and Models of Concurrent Systems*, NATO ASI Series **F-13**, New York, 1985, pp. 477–498.

[15] Harel, D., S. Efroni and I. R. Cohen, *Reactive Animation*, in: *Proc. 1st Int. Symposium on Formal Methods for Components and Objects (FMCO 2002)*, Lecture Notes in Computer Science **2852** (2003), pp. 136–153.

[16] Harel, D. and E. Gery, *Executable Object Modeling with Statecharts*, IEEE Computer (1997), pp. 31–42, also in *Proc. 18th Int. Conf. Soft. Eng.*, Berlin, IEEE Press, March, 1996, pp. 246–257.

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

[18] Harel, D. and Y. Setty, *Generic and Multi-Party Reactive Animation*, submitted .

[19] Heath, J., M. Kwiatkowska, G. Norman, D. Parker and O. Tymchyshyn, *Probabilistic Model Checking of Complex Biological Pathways*, in: C. Priami, editor, *Proc. Computational Methods in Systems Biology (CMSB'06)*, Lecture Notes in Bioinformatics **4210** (2006), pp. 32–47.

[20] Jensen, J., *Gene Regulatory Factors in Pancreatic Development*, Dev Dyn **229** (2004), pp. 176–200.

[21] Kam, N., D. Harel, H. Kugler, R. Marelly, A. Pnueli, E. Hubbard and M. Stern, *Formal Modeling of C. elegans Development: A Scenario-Based Approach*, in: C. Priami, editor, *Proc. Int. Workshop on Computational Methods in Systems Biology (CMSB 2003)* (2003), pp. 4–20.

[22] Kim, S. K. and R. J. MacDonald, *Signaling and transcriptional control of pancreatic organogenesis*, Curr. Opin. Genet. Dev. **12** (2002), pp. 540–547.

[23] Noble, D., *The heart is already working*, Biochem. Soc. Trans. **33** (2005), pp. 539–542.

[24] Plotkin, J., S. Prockop, A. Lepique and H. Petrie, *Critical Role for CXCR4 Signaling in Progenitor Localization and T Cell Differentiation in the Postnatal Thymus*, J. Immunol. **171** (2003), pp. 4521–4527.

[25] Priami, C. and P. Quaglia, *Modelling the dynamics of biosystems*, Briefings in Bioinformatics **5** (2004), pp. 259–269.

[26] Regev, A. and E. Shapiro, *Cellular abstractions: Cells as computation*, Nature **419** (2002), p. 343.

[27] Regev, A., W. Silverman and E. Shapiro, *Representation and simulation of biochemical processes using the pi-calculus process algebra*, in: *Pacific Symposium on Biocomputing*, 2001, pp. 459–470.

[28] Roux-Rouquié, M. and D. S. da Rosa, *Ten Top Reasons for Systems Biology to Get into Model-Driven Engineering*, in: *GaMMa '06: Proc. of the 2006 international workshop on Global integrated model management* (2006), pp. 55–58.

[29] Sadot, A., J. Fisher, D. Barak, Y. Admanit, M. J. Stern, E. J. A. Hubbard and D. Harel, *Towards Verified Biological Models*, IEEE/ACM Trans. Comput. Biology and Bioinformatics **to appear** (2007).

[30] Setty, Y., I. R. Cohen, Y. Dor and D. Harel, *Four-Dimensional Realistic Modeling of Pancreatic Organogenesis*, submitted .

[31] Swerdlin, N., I. Cohen and D. Harel, *The Lymph Node B Cell Immune Response: Dynamic Analysis in-silico*, submitted .

[32] Taubner, C. and T. Merker, *Discrete Modelling of the Ethylene-Pathway*, in: *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops* (2005), p. 1152.

[33] *TeleLogic,Inc*, www.ilogix.com.

[34] Turing, A. M., *The Chemical Basis of Morphogenesis*, Philos. Trans. R. Soc. London B **237** (1952), pp. 37–72.

[35] Turing, A. M., *The Chemical Basis of Morphogenesis. 1953*, Bull. Math. Biol. **52** (1990), pp. 153–197; discussion 119–152.

[36] Webb, K. and T. White, *Cell Modeling using Agent-based Formalisms*, in: *AAMAS '04: Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* (2004), pp. 1190–1196.

[37] Webb, K. and T. White, *Combining Analysis and Synthesis in a Model of a Biological Cell*, in: *SAC '04: Proc. of the 2004 ACM symposium on Applied computing* (2004), pp. 185–190.

[38] Webb, K. and T. White, *Cell Modeling with Reusable Agent-based Formalisms*, Applied Intelligence **24** (2006), pp. 169–181.