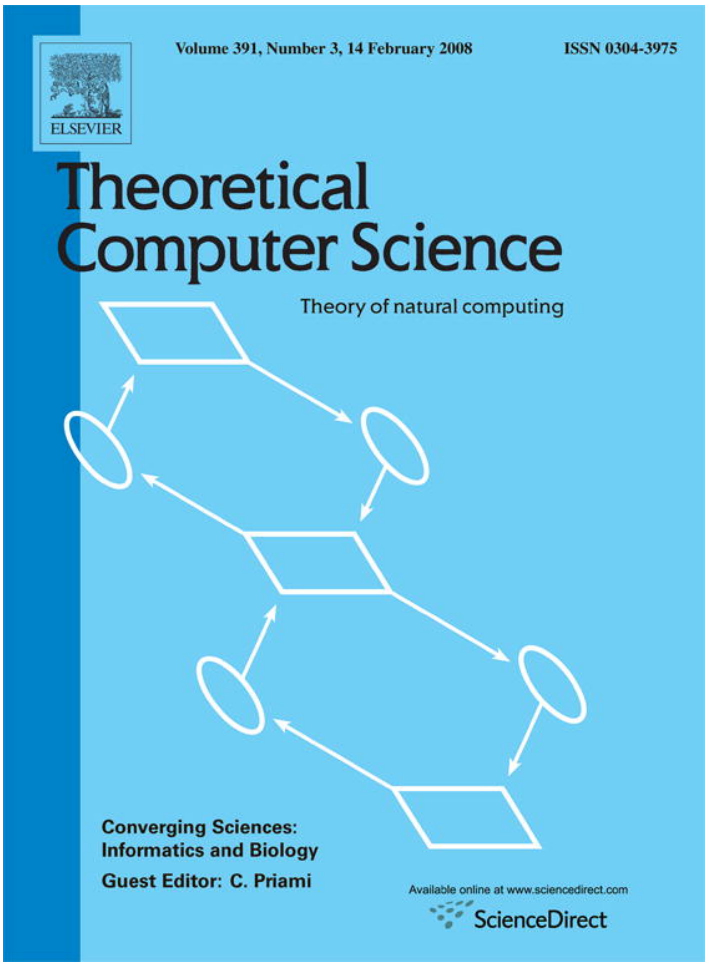
Volume 391, Number 3, 14 February 2008
ISSN 0304-3975
ELSEVIER
Theoretical Computer Science
Theory of natural computing
Converging Sciences: Informatics and Biology
Guest Editor: C. Priami
Available online at www.sciencedirect.com
ScienceDirect

Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 391 (2008) 276–290
Theoretical Computer Science
www.elsevier.com/locate/tcs

# GemCell: A generic platform for modeling multi-cellular biological systems

Hila Amir-Kroll[a,*,1], Avital Sadot[a,*,1], Irun R. Cohen[b], David Harel[a]

[a] *Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, 76100, Israel*
[b] *Department of Immunology, Weizmann Institute of Science, Rehovot, 76100, Israel*

**Abstract**

The mass and complexity of biological information requires computer-aided simulation and analysis to help scientists achieve understanding and guide experimentation. Although living organisms are composed of cells, actual genomic and proteomic data have not yet led to a satisfactory model of working cell *in silico*. We have set out to devise a user-friendly generic platform, *GemCell*, for Generic Executable Modeling of Cells, based on whole, functioning cells. Starting with the cell simplifies life, because all cells expresses essentially five generic types of behavior: replication, death, movement (including change of shape and adherence), export (secretion, signaling, etc.) and import (receiving signals, metabolites, phagocytosis, etc.). The details of these behaviors are specified in GemCell for particular kinds of cells as part of a database of biological specifics (the DBS), which specifies the cell properties and functions that depend on the cell's history, state, environment, etc. The DBS is designed in an intuitive fashion, so users are able to easily insert their data of interest. The generic part of GemCell, built using Statecharts, is a fully dynamic model of a cell, its interactions with the environment and its resulting behavior, individually and collectively. Model specificity emerges from the DBS, so that model execution is carried out by the statecharts executing with the aid of specific data extracted from the DBS dynamically. Our long term goal is for GemCell to serve as a broadly applicable platform for biological modeling and analysis, supporting user-friendly *in silico* experimentation, animation, discovery of emergent properties, and hypothesis testing, for a wide variety of biological systems.
© 2007 Elsevier B.V. All rights reserved.

*Keywords:* Biological modeling; Intercellular behavior; Executable modeling; Complex systems; Statecharts

## 1. Introduction

All living organisms are made of cells, so it would be desirable to have a single, generic tool to model and simulate all cells; describing biology generically by a single tool would allow biologists from different fields to communicate, to project knowledge from one biological field to another and to add continually emerging experimental data to already known biological facts. The desirability of deploying a generic simulation tool is clear; the rub is the complexity and

[*] Corresponding author. Tel.: +972 8 934 2961x2651; fax: +972 8 934 4103.
*E-mail addresses:* hila.amir-kroll@weizmann.ac.il (H. Amir-Kroll), avital.sadovski-sadot@weizmann.ac.il (A. Sadot), irun.cohen@weizmann.ac.il (I.R. Cohen), david.harel@weizmann.ac.il (D. Harel).
[1] Equal contribution.

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.11.014

diversity of cell systems. Could one design a single generic tool able to cope with cellular diversity that ranges from single prokaryotic cells to complex brains? This paper describes our design of such a generic cell tool — GemCell.

## *1.1. Biology needs computation*

Efforts to describe and study complex biological systems have yielded a vast amount of biological knowledge. This overwhelming and complex information has lead to the development of specialized branches of biological science. An appropriate merging of these branches could provide a far wider biological context. Thus, many acknowledge the need for computational models built for the purpose of integration and comprehension of this immense quantity of biological knowledge. Most studies present specific models built to describe a specific biological system or process, often one that goes on inside a single cell, or to answer a specific question [1–14]. However, a few have attempted a more generic approach. Simmune [15] is a software package built to provide a computational platform for modeling biological systems. The user of Simmune defines the molecular and cellular components of a system of interest and inserts relevant numerical values, such as association and disassociation rates of molecular complexes. The system then translates this data into mathematical equations, and the user can follow the simulation of the system on the GUI. The Virtual Cell [16] is a differential equations-based software environment designed for modeling cell biology. The user of the Virtual Cell can specify compartmental topology and geometry, molecular characteristics, and relevant interaction parameters. The software automatically converts the biological description into a corresponding mathematical system. The work presented in [17] describes a differential equations-based generic model of eukaryotic cell-cycle regulation. Another study [18] presents a generic model of chemotactic-based self-wiring of neural networks.

We believe that generalization of biological models is an important goal, and that initially, efforts should be invested in the integration of available biological information into comprehensive data. These data should be structured to describe the components of the biological system and the relationships between them. Moreover, the dataset should project on processes and describe the dynamics of these relationships (generally described *via* computational models). The combination of the described components and their dynamics would enable us to predict the system's function. Without understanding functionality, a complex system is little more than a collection of facts and details. The central question, we believe, is how to best combine generic and functional biology.

## *1.2. Genes and cells*

Two fundamental entities lie at the basis of life: the genetic code and its translation into functioning proteins, and the cell. The cell is the building block of all living organisms; the ability of the cell to carry out behavioral decisions in response to environmental stimuli enables – in fact defines – life. Cell–cell recognition, communication and mutual effects create a cell collective – an organism, or a colony of yeasts or bacteria. It has been claimed that the genetic code functions as the cell's program, nevertheless, the decoding of the genome [19] cannot predict the behavior of most cells, not individually and certainly not systemically [20]. Between the DNA sequence (genome) and the actual expression of specific proteins (proteome) lies a large knowledge gap, only little of which is known. Moreover, a very large knowledge gap lies between the proteome of a specific cell and the actual behavior of the cell. The collective behavior of many cells is particularly complicated. To a large extent, these knowledge gaps result both from our lack of essential biological information, and from our inability to merge the data that we do have into a fully functional and dynamic system. Indeed, at present it is not possible to reconstruct *in silico* a functioning cell bottom-up from real data about the genome, gene expression, signal transduction, metabolism, enzymatics, the proteome, and so forth. Construction of an entire multi-cellular system from intracellular data is out of the question.

## *1.3. GemCell*

In view of the above, we have decided to begin with the cell as a whole and not with the genome that formed it. We have adopted, as it were, a top-down point of view in which we relate to a whole cell as the smallest sub-unit structure of life. We envisage this cell as a "Lego piece" — a black box that makes its functional behavioral decisions by its interactions at the macroscopic level; at this stage, we have decided to defer trying to deal with intracellular components and their dynamics. By starting with the whole cell, we can approach a dynamic collective of many

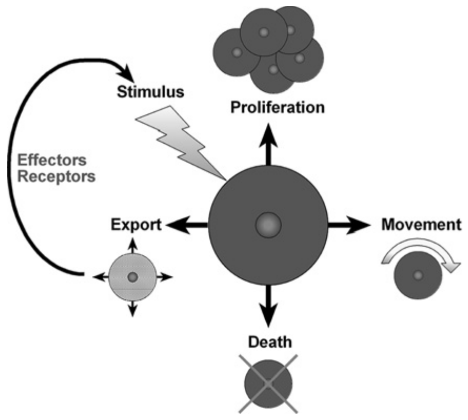

Fig. 1. Generic cell laws. Upon stimulation, a cell can behave in five different ways: it can *move*, *proliferate*, *export* information, *import* new information (receive signals) or *die*. Carrying out such behavior can lead to the import of new stimuli.

functioning cells in a way that will allow us to run extensive simulations and analyze the emergent outputs. Starting with the cell provides a means for exploring the behavior of complex biological systems *in silico*.

Our generic approach to the modeling, simulation and analysis of complex biological systems for the use of biologists, which we term *GemCell* (for *Generic Executable Modeling of Cells*), consists of four parts: (i) a generic state-based model of a functioning cell (including the cell's interactions with its surroundings), which is defined by generic biological laws, (ii) a database of biological specifics (DBS), which contains precise data about particular kinds of cells, molecules, environments and interactions, (iii) a means for connecting the generic laws with the DBS at runtime; this connection facilitates full executability of the model with the database enabling instantiation and changes in behavior of the generic model, and (iv) a means for visualizing the output of the run and for dynamic interaction of the user with the running model.

### *1.4. Generic cell laws*

We have constructed the generic base of GemCell by appreciating the fact that any cell, in response to its surroundings, carries out only five types of behavior (Fig. 1): *replication*; *death*; *movement* (including shape change and adherence); *export* (secretion of molecules, electricity, etc.) and *import* (receiving signals, metabolites, phagocytosis, etc.) [21]. The specific behaviors of any specific cell in response to stimuli determine the cell's fate and affect the emergent properties of its surrounding environment. The combination between these emergent environmental and cellular properties may lead to new stimuli that the cell and/or its neighboring cells can then receive. These resultant stimuli, in turn, lead to new behavioral actions of the receiving cells, and so on. Thus, generalization of biological processes can be achieved by the formation of a "web" of stimuli and responding cellular behaviors that evolves over time. The following sections elaborate these features of GemCell.

## 2. Methods

### *Statecharts*

Statecharts is a visual formalism, developed as a language for specifying reactive behavior [22]. The object-oriented version of the language that we use here allows one to define the behavior of objects [7,23], including the various states that an object can enter over its lifetime and the messages or events that cause its transition from one state to another. Statecharts describe both how objects communicate and collaborate and how they carry out their own internal behavior under different conditions. A statechart attached to a class specifies the behavior of all instances thereof.

### *Rhapsody*

Rhapsody is a development environment for the design of statechart-based models (see [23] and http://www.telelogic.com/products/rhapsody/index.cfm). In addition to providing a computerized visual design environment for performing object-oriented modeling, Rhapsody is capable of automatically translating any syntactically legal model into executable code (in C, C++ or Java). Once the model (or some part thereof) is

Table 1
The main classes and their associations in GemCell

| Class | Description | Associations |
|---|---|---|
| Controller | Defines the initial conditions of the system in the pre-run stage. Supervises time, creates and kills instances. | All classes (except Grid) |
| Environment (Env) | Defines the boundaries of the system. Holds the grid locations of the players in the system. Defines the system's chemical and physical conditions. | Controller, Cell, Molecules, Grid. |
| Cell | The smallest building block of life. Receives Signals and reacts by changing its behavior. | Controller, Env, Molecules, Signal, Export |
| Molecules[a] | Can be located in the Env or on a Cell surface (e.g., it can be a receptor) and, upon ligand recognition, can create a signal. | Controller, Env, Cell, Grid, Signal |
| Grid | Calculates the location and concentration of secreted molecules in the Env. Affected by molecular diffusion and consumption. | Molecule, Env |
| Signal | Defined by a specific cell, a receptor and its ligand, and their recognition level. Dictates Cell behavior. Can create Exports. | Controller, Cell, Molecules, Export |
| Export | Defines the Molecules that can be exported by a Cell as dictated by a specific Signal. | Controller, Cell, Molecules, Signal |
| Cell Interior | Not in the project at this point | |

a We define Molecules as a *collection*; a cluster of molecules on the grid; we do not deal with single molecules.

constructed and translated into code, Rhapsody can execute it so that one can observe its progress, e.g., in animated versions of the model's statecharts, or linked to a more realistic GUI. Animation shows the current states and transitions by coloring them uniquely.

*MySQL*

MySQL is a relational database management system. It supports the Structured Query Language (SQL), which is used for adding, accessing, and processing all data in the database. MySQL is open source (see http://www.mysql.com).

## 3. GemCell structure

### 3.1. Model structure

Our generic cell model was constructed to dictate the main laws of cell–cell and cell–environment interactions, followed by cellular response. It is built using the language of *Statecharts* [22], and we have implemented it using the *Rhapsody* tool [23] (http://www.telelogic.com).[2] To carry out the modeling, we determine the main generic classes of cellular interactions and responses and model them in a class/object model diagram. See Table 1 and Fig. 2. The classes are built in three levels: (i) the Controller,[3] which is responsible for system maintenance and time supervision, (ii) the Environment (Env), Cell and Molecules classes, which represent tangible entities with relevant attributes and operations, and (iii) the Signal,[4] Grid and Export classes, which represent abstract entities, such as interactions, actions and behaviors. Each generic class can represent a wide variety of biological entities. Thus, for example, a Cell symbolizes all possible types of cells; e.g. T-cells, liver cells, etc.

### 3.2. Dynamic behavior

To achieve generic modeling of dynamic processes, we describe the fundamental generic events of cellular interactions and the resulting behavior. Using statecharts, we constructed chains of possible biological trees of events; thus, we follow the trail leading from the formation of a new cell (Cell) and its receptors (Molecules), to ligand (Molecules)-recognition by the cell's receptors, to the import of a signal (Signal), to a chosen cellular behavior dictated by the collection of the obtained signals (Signal) and to the resulting action (*move*, *proliferate*, *export, import*, *die*) of the cell (Cell). Newly exported molecules are then dispersed on the grid (Grid). The cells' receptors can then

[2] The model could have been implemented in any of the several tools that support Statecharts and their full executability, such as Stateflow (http://www.mathworks.com/), RoseRT (http://www.ibm.com) or StateRover [24].

[3] We represent *class* by using Capital letters and a distinct font (e.g., Cell).

[4] Signal — the actual massage of a stimulus.

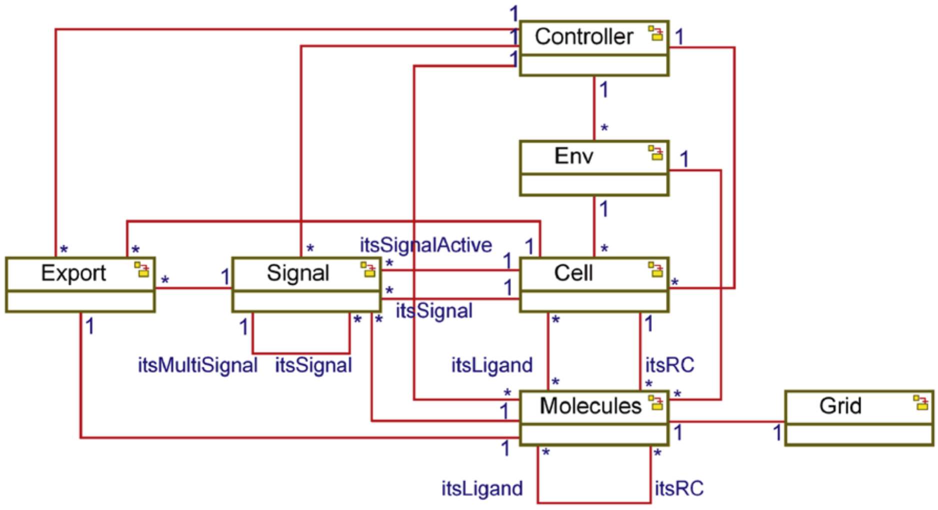

Fig. 2. Object Model Diagram. Olive-colored squares represent Classes; red lines stand for associations[5] between Classes; blue numbers depict how many instances of a class are associated with the connected class (e.g., one Environment is associated with many (*) instances of Cell); blue names of associations appear when more than one kind of association exists between classes (e.g., a Cell can recognize Molecules as its ligands or as its receptors (RC, Receptor Collection)).

recognize new ligands, form new signals, and so forth. The web of signals→behavior→new signals thus evolve through time.

## 3.3. The database of biological specifics (DBS)

We hold that generalization of biological processes *per se* is not worth much, unless utilized to describe and understand actual complex biological systems. For the purpose of collecting and describing specific biological information, we have designed a database, constructed using MySQL. As explained below, the DBS links to the generic statechart model, so that specific runs of GemCell are achieved by the dynamic interaction between the two.

*Database structure*

The database is organized in a tree-like structure composed of four layers of tables, each layer relying on information from previous layers (see Fig. 3). The top layer consists of tables that represent tangible entities (the Cell, Env and Molecules). The other layers consist of tables that represent abstract entities, such as interactions and actions.

The tables contain three types of data: type I supplies information to the generic platform to enable a specific run, type II defines animation parameters for the visualization of the output (such as color and shape), and type III consists of data that is of interest to the experimentalist (such as molecule description). For detailed description of the database tables, including samples of real biological data insertions, see Tables 2–7.

## 3.4. Executing generic and specific events in tandem

As mentioned above, to facilitate actual runs (executions) for the purpose of performing specific experiments, the generic Statechart model interacts with the DBS on the fly (Fig. 4). During a run, the various parameters of each instance of the classes, Cell, Molecule, Env, Signal, etc., update their actual values from the database. In this way, the differences between various executions of the model are rooted in the specific biological data extracted from the database. GemCell can thus be viewed as consisting of a general mold, which upon interaction with a specific dataset results in variable outcomes. In other words, GemCell emulates the wide variety and flexibility of outcomes one finds in real life.

*Model execution*

Model execution is carried out in two stages. In the *pre-run* stage, the user defines the participants that can 'play' (participate) in the run. This definition includes the identity of the participants and relevant initial conditions. At the

[5] Association defines a semantic relationship between two or more classifiers that specify connections among their instances.

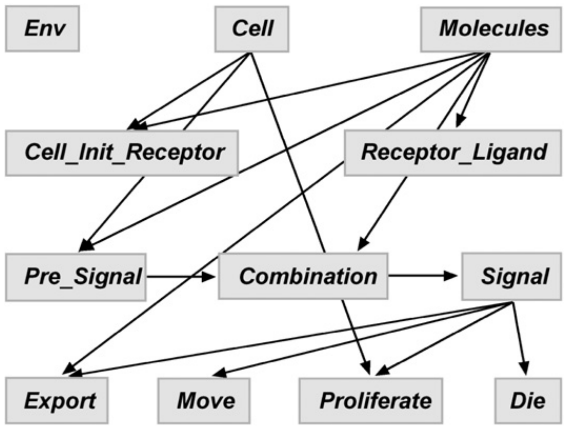

Fig. 3. Database scheme. Arrows demonstrate the information route between tables. Information between tables is delivered only by the unique ID key of each record in the delivering table. The first layer defines the main participants in the model: *Env* (the environment), *Cell* and *Molecules*. The second layer defines the relationships between the main participants: the *Cell_Init_Receptor* table defines the receptor set of newly-formed cells, and the *Receptor_ Ligand* table defines recognition between molecules. The third layer defines the process of formation of signals: the *Pre-Signal* table defines the formation of a signal, dictated by the combination of a specific cell, receptor and ligand and the recognition level between them. The *Combination* table defines new signals that are created by combining several signals. The *Signal* table defines the final signals and the basic behaviors, detailed in the next layer. The fourth layer is behavioral. It defines in detail the possible behaviors that a cell can perform as a result of a signal: *Export*, *Move*, *Proliferate* and *Die*.

Table 2
*Cell*[a]

| Cell_ID | Name | Status | Half_Life | Speed | Size | Shape |
|---|---|---|---|---|---|---|
| 1 | nDC | Naïve | 48 | 4 | 4 | Dendrite |
| 3 | eTh1 | Effector | 100 | 4 | 2 | Round |
| 5 | nTh | Naïve | 504 | 4 | 1 | Round |
| 7 | aTh | Active | 72 | 4 | 2 | Round |

The *Cell* table is in the top layer of the database. It describes specific parameters of the various cells of a specific, well-defined biological system. In this example, the cells are part of the immune system, and participate in the inflammation process. Each cell is described by a unique ID number (**Cell_ID**), which is the key field of the database, and by a **Name** composed of its scientific name (e.g., **T h**elper cell would be called **Th**) and its **Status** (e.g., the Th cell is in the **n**aïve status and therefore would be named **nTh**). Similarly, the cell nDC is a **n**aïve **d**endritic **c**ell. **Half_Life**, **Speed**, **Size** and **Shape** represent parameters that are used by the generic model.
[a] Due to lack of space, we show only samples of the relevant fields and data for this and the following tables.

outset, the user determines one or more biological environments, e.g., an organ, a tissue or an *in vitro* experimental environment, such as a Petri dish. These environments must have been pre-defined by the user in the database (Env table; data not shown). The user then populates the chosen environment with cell populations and determines the types and amounts of the participating cells. GemCell then distributes the cells automatically within the borders of the defined environment. The user then determines the type, concentration and location of initial molecules that are placed in the environment or that are secreted by the cells.

The main stage, the *run*, starts when all participants have been formed, have retrieved their initial parameters from the database and are ready to 'play'. The generic Statechart model then causes cell populations to perform cycles of signals→behavioral-response→new signals, leading to a functional overview of the system and to biological outcomes.

*Time*

Our approach captures time as discrete, and composed of time units. *Time-unit* length (e.g., hours, seconds, etc.) is determined by the user. In computer science terms, our model is synchronous, meaning that cells may collect signals and act accordingly only once per *time unit*. Only when all cells have finished receiving and acting upon their signals, next *time unit* will be carried out (Fig. 5). The special Controller is responsible for time management. We feel that discrete time captures the discrete behavioral states we envision when we study experimentally discrete cells.

Table 3
*Molecules*

| Molecule_ID | Name | DB_ID | BC | RC | EC | Diffuse_Rate |
|---|---|---|---|---|---|---|
| 39 | CD4 | 920 | 1 | 1 | 0 | 5 |
| 27 | CD40L | 959 | 1 | 1 | 0 | 5 |
| 72 | CXCR3 | 2833 | 1 | 1 | 0 | 5 |
| 66 | HSP60 | 3329 | 0 | 0 | 1 | 5 |
| 53 | IL-2 | 3558 | 0 | 0 | 1 | 5 |
| 41 | IL-2R | 3559/60 | 1 | 1 | 0 | 5 |
| 42 | IL-2R' | 3559/60/61 | 1 | 1 | 0 | 5 |
| 31 | MHC-II | 3122 | 1 | 1 | 0 | 5 |
| 58 | MHC-II(i) | 3122 | 1 | 1 | 0 | 5 |
| 89 | MIG | 4283 | 0 | 0 | 1 | 1 |
| 59 | TCR(i) | 6955/7 | 1 | 1 | 0 | 5 |

The *molecules* table is on the top layer of the database and describes parameters of molecules. Each molecule is tagged with a unique ID key number (**Molecule_ID**) and with its scientific **Name**. The parameters **BC**, **RC** and **EC** are Booleans. The letter C represents a **c**ollection of molecules. **BC** represents **b**ounded molecules, namely ones that are bound to the cell surface. **RC** represents **r**eceptors and **EC** represents secreted **e**ffectors, namely molecules that were exported to the environment. The parameter **Diffuse_Rate** represents the rate at which a collection of molecules is expected to diffuse. It is valued on a discrete 0 to 10 scale. The **DB_ID** parameter describes the ID number of the molecule in other databases. This parameter represents a different type of information in the database, a type that supplies data to the experimentalist rather than to the generic model. In this example the **DB_ID** is taken from the Gene database (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=gene). In the example presented here, the presented molecules are part of the inflammation process and their **Molecule_ID** number is used by the following tables (see below).

Table 4
*Pre_Signal*

| Signal_ID | Cell_ID | Receptor_ID | Ligand_ID | Temp | pH | Time |
|---|---|---|---|---|---|---|
| 10 | 1 | 0 | 0 | >39 | | |
| 16 | 5 | 59 | 58 | | | |
| 39 | 5 | 39 | 58 | | | |
| 61 | 1 | 58 | 59 | | | |
| 92 | 3 | 72 | 89 | | | |

The *Pre_Signal* table is on the third layer of the database and holds data that defines the Signals. Each signal consists of the **Cell_ID**, which is provided by the *Cell* table, and the **Receptor_ID** and the **Ligand_ID,** which are provided by the *Molecules* table. Other signal-effectors can be environmental parameters such as temperature (**Temp**) and **pH**, or **Time**. The parameter **Signal_ID** is a unique key and is utilized by the *Signal_Combination* table (Table 5). For example, receptor #59 (TCR(i)) on cell #5 (nTh) can bind ligand #58 (MHC-II(i)) to form signal #16. Note that this ligand can be also bound by another receptor, #39 (CD4), on the same cell.

Table 5
*Signal_Combination*

| Final_Signal_ID | Pre_Signal_ID |
|---|---|
| 10 | 10 |
| 16 | 16 |
| 39 | 39 |
| 55 | 16 |
| 55 | 39 |
| 77 | 61 |
| 110 | 92 |

The *Signal_Combination* table holds all possible combinations of signals. The combination of two or more signals forms a multi-signal. In this example, **Pre_Signal_ID** #16 and **Pre_Signal_ID** #39 can be combined to form a new multi-signal, with a **Final_Signal_ID** of #55. If the two signals are imported together by the nTh cell (*Cell* table; **Cell_ID** #5), they are both ignored and the cell imports the corresponding multi-signal. Accordingly, other signals also receive new **Final_Signal_IDs**. This parameter is the key number of the signal for both the database and the generic model.

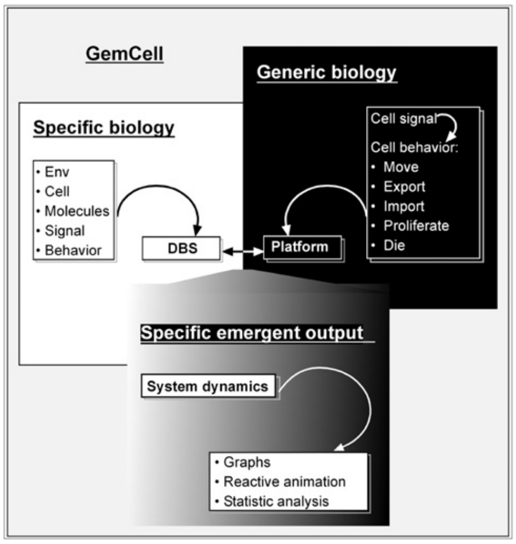

Fig. 4. Structure of GemCell. The GemCell model is a combination of generic biology, formed by five generic laws of intercellular behavior, and specific biology, built from the data of actual cellular systems. GemCell consists of three components: the generic state-based model, which models the generic biology; the DBS, which holds the information about the specific cellular system that the user wishes to model; and the output of the system's dynamics that emerges from the execution of the generic model combined with the DBS. These three components are integrated and operate together continuously during model execution.

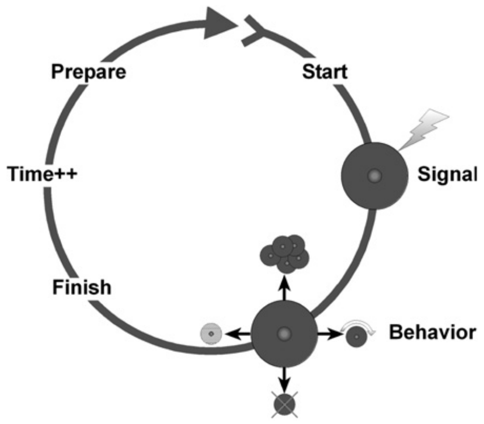

Fig. 5. Time. We monitor time in a discrete synchronous fashion. For each time unit (days, hours, seconds, etc.) the model collects all the signals and the possible actions of each cell in the system and proceeds to carry out the required behavior. When all cells have finished receiving their signals and acting upon them, the model progresses to the next time unit, allowing new signals to be imported. New signals are formed as a consequence of the behavioral outputs of the cells and the resulting emerging properties of the environment. Thus, we form a web of signals and their corresponding cellular behaviors that spreads through time.

## *The Cell statechart*

We now describe in some detail the statechart of the Cell class, the main participant in the generic model (Fig. 6).

At the pre-run stage, or due to proliferation in the run stage, a Cell instance is formed [Form].[6] The newborn cell queries the DBS to obtain from the *Cell*[7] table (Table 2) the actual values of its parameters. For example, if the user has determined that the particular Cell instance's type is a 'naïve T helper' cell of the immune system, then its **Cell_ID**[8] number (unique for this cell type) would be 5, its **Name** would be nTh, its **Size** would be 1 (one square on the grid), etc. Hereafter, the cell will communicate with the database *via* this **Cell_ID** number.

The cell then enters the [Form.CreateRC] state (encapsulated within state [Form]; Fig. 6B), in order to create a set of receptors that are common to a specific cell type and that are in fact part of the cell's type definition. The cell

6 [Representation of a *state*].
7 *Tables* are written in an italic format.
8 Bold letters represent an attribute of a class.

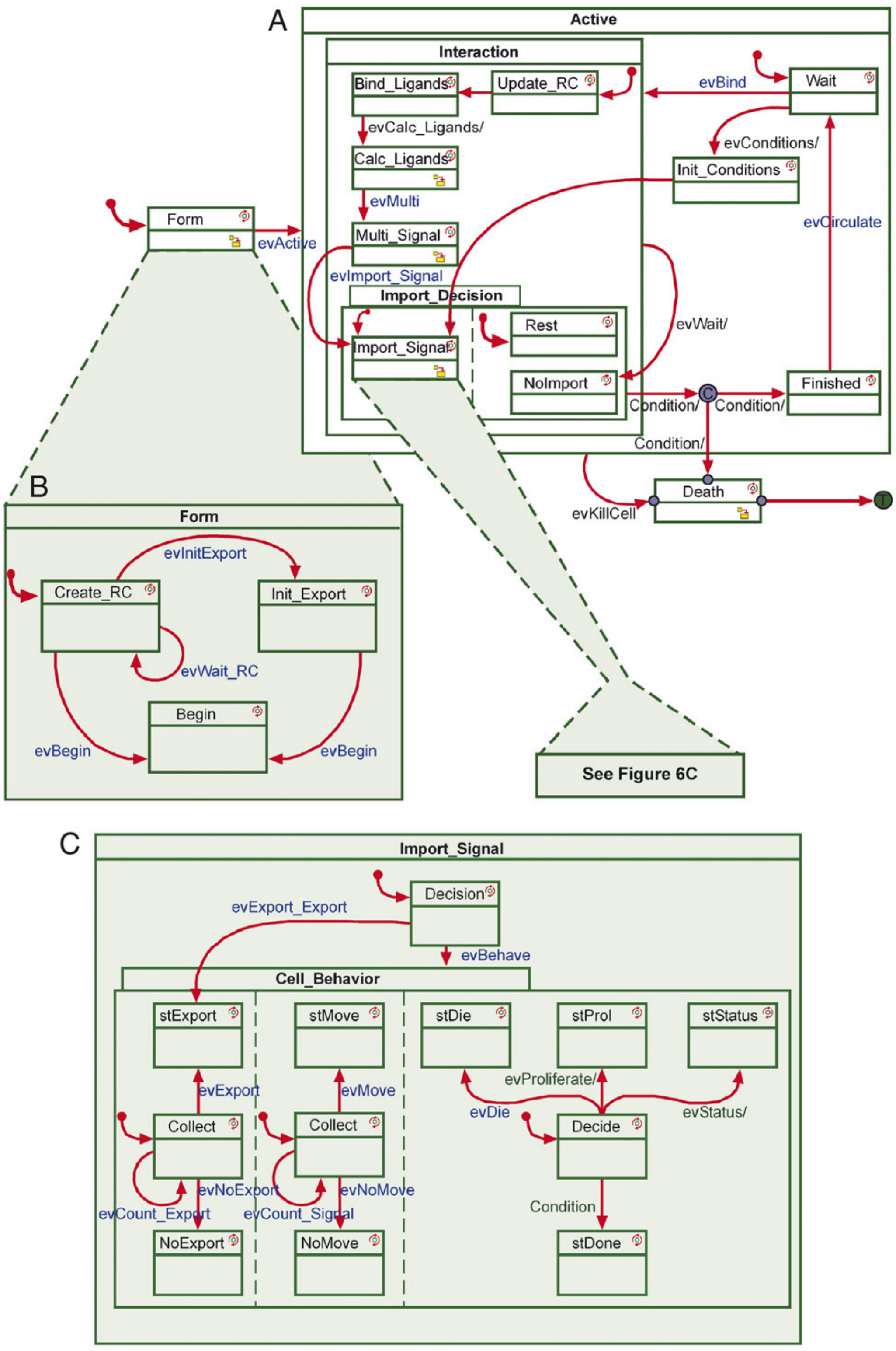

Fig. 6. A. Statechart of Cell. Green boxes represent states, in which a Cell can be at, during its lifetime. The green dashed line within the [ImportDescision] state represents a state's partition into orthogonal states – and situation, in which the two different processes may occur simultaneously. Red lines are transitions between states. The letter C inside the blue circle indicates a condition junction. The letter T inside the green circle indicates a termination state. The text contains explanations about the various states of the Cell. B. Sub-statechart of [Form]. The sub-statechart encapsulated in the [Form] state of the Cell statechart. For details see text. C. Sub-statechart of [Import_Signal]. The sub-statechart encapsulated in the [Import_ Signal] state of the Cell statechart. It shows the various behaviors a Cell can preform during its lifetime. Note that due to orthogonal states a Cell can [Export] and [Move] simultaneously, as well as [Die] or [Proliferate] or change its [Status], but it cannot do all these at the same time.

utilizes its **Cell_ID** number to request its receptor list from the *Cell_Init_Receptors* table (not shown) in the database. The relevant data include the receptors' **Molecule_ID**, **Name**, **Concentration**, etc. With this information, the cell instance creates new instances of receptors that belong to the class Molecules. In turn, these newborn instances of receptors (Molecules) go to the *Molecules* table in the database (Table 3), armed with their **Molecule_ID** number and request additional relevant data, such as their **Diffuse_Rate**. More information is requested from the *Receptor_Ligand* table (not shown). This table holds information regarding the recognition between a certain receptor and its ligands, e.g., the **Affinity** between them.

In the next step, the cell enters the [Form.InitExport] state, in which it secretes relevant molecules to its environment. This secretion involves a process similar to the one described for the receptors creation. The exported molecules are diffused on the grid (see diffusion calculation in the text). The cell instance continues to the next state [Active], which defines the cell's time-unit cycle of receiving and responding to signals. In the state [Activate.Wait.RandomMove] ([RandomMove] state is encapsulated in state [Wait]; data not shown), the cell calculates its location on the Env grid, and then goes to [Activate.Interaction], in which it receives signals and responds accordingly.

When all cells are formed, in the pre-run stage, the user can start the run stage. The cell enters the [Activate.Interaction.Update_RC] state, in which it receives a list of possible ligands from its receptors. The Env holds lists of all the molecules that were created during the run, and their concentration and location on the grid. The cell then enters the [Activate.Interaction.Find_Ligand] state, in which it 'asks' the Env if any of the ligands in its ligand list are present in its surrounding (the nine squares around it on the grid). If relevant ligands are present, the cell calculates the recognition level between these ligands and their associated receptors at the [Activate.Interaction.Calc_Ligands] state. The recognition level depends on the concentrations of both ligand and receptor and on the affinity between them. If recognition is above a certain threshold, the cell creates new instances of Signals. The signal instance is defined by a specific cell, a specific receptor and a specific ligand, and is scored for the recognition level between the receptor and its ligand (Table 4).

If several signals are received simultaneously by a cell, they can be combined to create a multi-signal. The cell moves to the [Activate.Interaction.Multi_Signal] state, where it checks the table *Signal_Combination* (Table 5) for possible combinations of the individual signals in its list that can be grouped to form a multi-signal. If so, it will create a new multi-signal instance of the Signal class. The individual signals that compose the multi-signal will not be imported individually by the cell.

The cell then moves to the [Activate.Interaction.Import_Decision. Import_Signal] state, in which it imports all viable signals, ordered from high to low by their recognition level score, and executes them using a queue. Encapsulated in this state are the states that participate in the actual behavioral execution (Fig. 6C). Signals are now in their [Decision] state (Signal statechart; data not shown) in which they retrieve data from the *Signal* table (Table 6) concerning their effect on the cell. Accordingly, the signals request relevant data from the four possible behavioral tables: *Export* (Table 7), *Move*, *Proliferate* and/or *Die* (not shown). The information in these tables specifies the actual behavior in detail. For example, a moving cell can be attracted to, or repelled by, a molecular gradient (chemotaxis), it can adhere to a neighbor cell, it can roll, etc. The cell waits in the [Active.Interaction.Import_Signal.Decision] state until the suggested behaviors from all signals are collected. When this has been done, the cell executes the collected behaviors by transitioning to the relevant states.

Once the cell has finished this cycle of execution, it leaves the [Interaction] state and continues to the [Active.Finished] state. The cell informs the Controller that it has finished the current cycle and transitions to the [Active.Wait] state. When all cells have completed their behavioral cycle, the Controller advances time, and the activity of the next time unit can start. The cells re-enter the [Active.Interaction] state, and the cycle repeats.

*Discrete perception*

To support biological usage and understanding of the model, we describe quantitative data in a discrete fashion. In this way, quantitative data inserted into the database by experimental biologists (e.g., concentration, affinity) can be relative and follow a simple scale of 0–10. This scale can represent a logarithmic or an arithmetic scale. GemCell will make use of this quantitative data during the run, carrying out minor needed calculations dynamically. Two examples are detailed below.

Diffusion: The diffusion of molecules in the environment depends on their concentration and on their diffusion rate. When a cell exports molecules to the environment, the molecules are secreted to the nine grid squares surrounding the cell. Thus, each instance of the Molecules class holds an instance of a grid with the information of the molecules'

Table 6
*Signal*

| **Signal_ID** | **Export** | **Move** | **Proliferate** | **Die** | **New_Status** | **New_ID** | **Time_0** | **Est** |
|---|---|---|---|---|---|---|---|---|
| 10 | 1 | 0 | 0 | 0 | | | | |
| 16 | 0 | 0 | 0 | 1 | | | | |
| 39 | 0 | 1 | 0 | 0 | | | | |
| 55 | 1 | 1 | 1 | 0 | active | 7 | 2 | 9 |
| 75 | 0 | 0 | 0 | 1 | | | | |
| 77 | 0 | 1 | 0 | 0 | | | | |
| 110 | 0 | 1 | 0 | 0 | | | | |

The *Signal* table is in the third layer of the database and holds the behavioral aspects of the Signals. The **Signal_ID** parameter is in fact the **Final_Signal_ID** from the *Signal_Combination* table. The parameters **Export**, **Move**, **Proliferate** and **Die** are Booleans. These parameters sum the basic behavior of a given signal and point to the next layer of behavioral tables. For example, **Signal_ID** # 10 points to further information in the *Export* table and **Signal_ID** # 55 points to further information in the *Export*, *Move* and *Proliferate* tables. A signal can also indicate a status change, in which a cell can change its status, but not its identity. In this example, **Signal_ID** # 55 imposes a status change on the naïve nTh cell, so its **New_Status** is active, and its **New_ID** number is 7. The **Name** of this cell will be consequently changed to aTh (see the *Cell* table). However, a T cell can never change into a dendritic cell. The **Time_0** parameter counts the time units between the initiation of an import of a signal and the execution of one of its resulting behaviors (in this particular dataset, a time unit equals one hour). In the example, nTh becomes activated within two time units after the signal was imported. The **Est** parameter represents a 0 to 10 scale, in which the user can estimate the certainty of a given parameter (0 = very certain, to 10 = uncertain). In this example, the estimation of **Time_0** = 2 time units was estimated to be **Est** = 5, meaning that this data has quite low credibility. Hence, if the specific run based on this dataset fails to imitate real life, the **Time_0** = 2 value will become a candidate for modification.

Table 7
*Export*

| **Signal_ID** | **Molecule_ID** | **Conc** | **Est** | **Time_0** | **Est1** | **Time_$\Delta$** | **Est2** |
|---|---|---|---|---|---|---|---|
| 10 | 66 | 7 | 1 | 1 | 0 | 1 | 5 |
| 55 | 53 | 9 | 1 | 1 | 0 | 92 | 10 |
| 55 | 41 | 0 | 0 | 0 | 0 | 1 | 0 |
| 55 | 42 | 9 | 1 | 0 | 0 | 1 | 0 |

The *Export* table is on the fourth layer of the database, and is an example of a behavioral table. It holds the data regarding exports of molecules. The **Signal_ID** is derived from the *Signal_Combination* table and from the *Signal* table. The **Molecule_ID** parameter indicates which molecule the cell should export as a consequence of a given signal. The cell exports these molecules in a certain concentration, represented by the **Conc** parameter, which is a discrete 0 to 10 scale. The **Time_$\Delta$** parameter counts how many time units the export should last. In this example, **Signal_ID** # 55 is responsible for the export of **Molecule_ID** # 53 (IL-2) from an **nTh** cell. **Molecule_ID** # 53 is exported in high concentration (**Conc** = 9), starting after one time unit (**Time_0** = 1) and lasting for 92 time units (**Time_$\Delta$** = 92). The user is very uncertain regarding the **Time_$\Delta$** value (**Est2** = 10), but quite certain regarding the concentration (**Est** = 1). Note that **Molecule_ID** # 41 has a **Conc** value of 0, meaning it should be deleted from the cell repertoire (in which case the IL-2 receptor in a certain form declines and another form of IL-2 receptor, molecule # 42, takes its place on the cell surface).

concentration on each square thereof. At each passage of a time unit, the molecules diffuse further to the environment, and their concentrations on the squares of the grid are updated.

The diffusion itself is calculated as follows: every square on the grid holds a certain concentration of a given molecule, from which it 'contributes' a small portion to each of its surrounding eight squares. This portion is calculated by multiplying the concentration present on that specific square by the diffusion rate of the given molecule. Hence, although a grid square 'loses' some of the amount of the molecule to its neighbors, it may also receive a certain amount of that molecule from its neighbors. Naturally, a cell can export a certain molecule for the duration of several time units, and it may move while exporting. In such a case the amount of the secreted molecule is added to the existing grid instance, taking into consideration the changed location of the secreting cell. Moreover, each type of molecule can be secreted by several cells, but it is represented by a single instance of Molecules, that holds only a single instance of Grid.

Chemotaxis: A cell can move on the grid either in a random fashion or in response to specific signals. The signals may cause movement towards an attractant or away from a repellent. A cell can move to any of its eight surrounding squares on the grid (limited by the boundaries of its environment), or it can choose not to move. In every time unit, the cell calculates the probability of each of these nine possibilities. If there is no attractant or repellent in the cell's surroundings, or if it has not received any specific signal to move in a certain direction, the possibilities that all have equal probability (1/9), resulting in a random move. If there is an attractant (repellent) present, there is a higher (lower) probability that the cell will move in its direction and lower (higher) probability that it will move in other directions. If there are several attractants or repellents present in the surroundings of the cell, the probabilities of the movement of the cell are calculated according to the recognition level of the cell and these ligands (derived from the appropriate signal) and by the actual concentration of each ligand in each of the squares around the cell. The higher the recognition level is, the higher the chance the cell will be attracted to or repelled from a given ligand.

*Probability*

GemCell uses probability in various places and in different ways, and thus enables the model to produce various outcomes from different runs with the same dataset. Moreover, it can expose phenomena that are statistically rare.

In the database:

- Variable choice of amounts, concentrations, etc. For example, some cells of a certain type present a certain receptor in a high concentration while others in a low concentration.

In the pre-run stage:

- Random location of cells on the grid of a given environment. This creates diverse interactions among cells, as well as between cells and substances present in the environment.

During the run:

- Random movement of cells on the grid per time unit. Cells are allowed to move in varied directions and distances on the grid, depending on their speed (e.g., immune cells are more “frantic” than tissue cells).
- Random import of signals with an equal recognition level.
- Probabilistic movement of cells in response to gradients of molecules, such as chemokines (for details see above).
- Random order of the cells in the execution queue.
- Variable ligand consumption. (The first cell in the queue to consume a certain ligand will probably have more ligand to consume than its neighboring cells.) The consumption is defined by the amount of possibly consumed ligand (calculated by the recognition level between the receptor and its ligand), but limited by the actual ligand concentration around the specific cell (the lower of the two).

### 3.5. *Visualization and animation*

During a GemCell run, the emergent properties of the biological system being executed should be presented coherently and dynamically, so that the user is able to follow the computational experiment, ask questions of interest and interact with the run. For this purpose, the user may automatically export chosen results to output files. The information is collected in these files per time unit and can be presented in graphs *via* several possible software tools, such as Matlab and Excel (see Fig. 7).

In the future, we will incorporate the technique of *reactive animation* [25], where GemCell will drive an interactive animation of the run produced by state-of-the-art animation software. As part of this, we plan to support the biologist’s point of view by animating cell behavior in a way similar to microscopic images. Using reactive animation, the experimentalist will be able to interfere with and dictate the run by controlling components of the biological system.

## 4. Discussion

### 4.1. *Intuitive usage*

The overall goal of this work is to build a generic tool usable by experimental biologists. This dictated our perspective throughout the design and implementation of GemCell, which we built as an engine-like machine, whose use does not impose much technical understanding. Nevertheless, the rational for the design of the dynamics embodies in the states and transitions of the Statechart model is based on simple biological generic laws. Moreover, we focused on biological functionality rather than on the exact quantitative biochemical and physical parameters. It is no surprise, therefore, that the result is a discrete description of the biological system. Indeed, the use of a discrete and relative

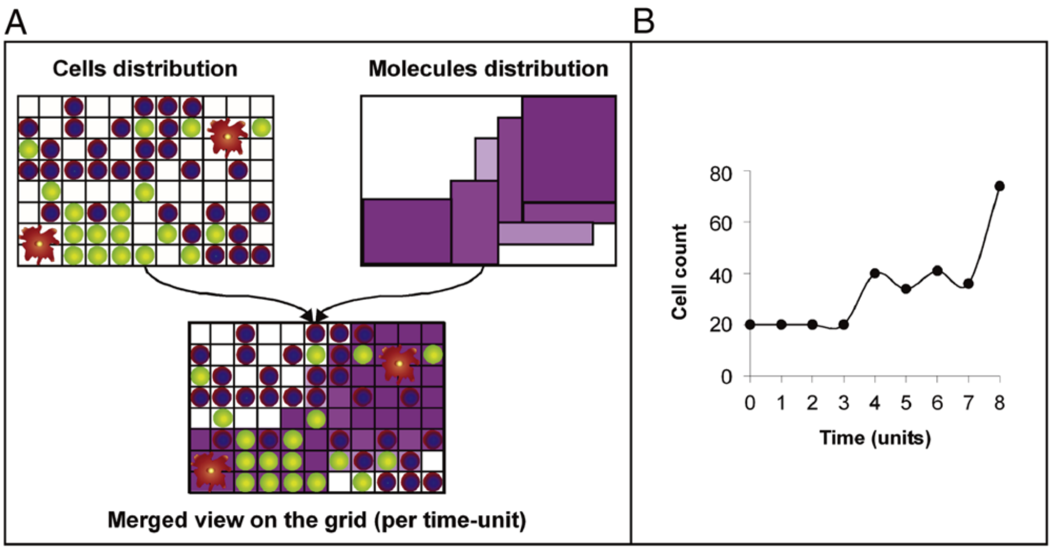

Fig. 7. A. An example of a possible output single time unit. The top left panel shows an example of cell distribution on the grid. We can see three types of cells: a blue cell and a green cell, each of size one grid square, and an orange cell of size four grid squares. Cells that are adjacent may interact with each other and behave accordingly. The top right panel shows an example of distribution of molecules on the grid. The gradient of the color indicates levels of concentrations of the molecules on different grid locations. The bottom panel shows a combination of the two top panels. Cells that are surrounded by molecules can interact with them, and then behave accordingly. For example, they can move towards a gradient. In this example, the indicated molecule was exported by the orange cells. B. An example of a possible output over time. This graph shows the change in the number of cells in a population over time. The changes occur as a result of proliferation and death signals.

spectrum of concentrations, affinity and time seems to be an appropriate way to deal with the difficulty of performing exact quantitative measurements of biological dynamics. In any case, biologists think discretely and experiment discretely.

Having said that, it is important to realize that the DBS is the main connection between the potential user (the biologist) and GemCell. Accordingly, the database is designed to capture the data needed for actual biological functionality and by complementing the generic dynamics present in the Statecharts. In addition, the database is designed to support discretization of biological parameters.

We have tried to build the database and its tables in an intuitive fashion, for the benefit of potential users, who can insert their data of interest directly to form their own set of data, or to combine it with existing data inserted by others. We plan to make this even easier in the future by designing a more flexible and intuitive interface for the end-users. Such a tool can ease data insertion to the database and reduce to a minimum, errors that might result from manual filling of the tables. Once the data has been inserted into the database, the user can run as many experiments as desired. Since the validation of inserted data can be scored by the user during insertion (**Est** parameter, see Table 6), questionable parameters can be adjusted. Of course, some kinds of results emerging from the model's execution can be viewed as predictions or hypotheses and would need to be tested by laboratory experiments. Once verified these can then be incorporated back into the database and used for further *in silico* experiments.

We are currently in the process of applying GemCell to a number of specific well-understood biological systems. This gives rise to a learning process by which GemCell benefits from fine-tuning of both the generic statechart model and the DBS structure.

## *4.2. Summing up*

The prospective benefits of GemCell, we believe, are four:

(i) Organized database. To be accessible, the database needed to run GemCell will oblige the biologist to put his or her house in order. Data will have to be transformed into precise and organized knowledge; the scientist will become acutely aware of any lack of key data.

(ii) Realistic representation. The realistic representation of GemCell, as has been shown in Reactive Animation [25], will trigger fruitful associations in human minds.

(iii) Emergent properties. The dynamic integration of heretofore-isolated experimental facts can disclose emergent properties of the system hidden in the separated details [26]. Indeed, the transformation from single cell properties, as defined in the DBS, to the massive emerging output of the cell collective, i.e., to the environmental level, can reveal the function of the cell collective.

(iv) Experimentation. The recognition of emergent properties and the new associations that arise from the realistic representations of GemCell will stimulate new ideas. New ideas lead to new experiments, both *in vivo* and *in silico*.

While GemCell is still in its early stages of evolution and has not yet been tested widely, we hope that it will eventually contribute to a better understanding of biological systems and their dynamics. By using a DBS linked to a generic dynamic tool for cell behavior the introduction of new data into the model will be relatively easy, and will be done by the experimentalist, with no need to remodel additional knowledge by an expert. Moreover, utilizing generalization of functional biology can enable researchers from different fields to use GemCell and benefit from a fruitful biological cross-talk.

## Acknowledgements

We wish to thank Dr. Yoram Louzoun and Dr. Guy Shakhar for their important assistance in drawing biological rules, and to Aron Inger, Shahar Maoz and Asaf Kleinbort for their computational help.

This work is supported in part by the John Von Neumann Minerva Center for the Development of Reactive Systems at the Weizmann Institute of Science.

This work is supported in part by a grant from the Kahn Fund for Systems Biology at the Weizmann Institute of Science.

Part of D. Harel's work was carried out during a visit to the School of Informatics at the University of Edinburgh, which was supported by a grant from the EPSRC.

H. Amir-Kroll is partially supported by the Yeshaya Horowitz Association through the Center of Complexity Science.

## References

[1] J. Barjis, I. Barjis, Formalization of the protein production by means of petri nets, in: International Conference on Information Intelligence and Systems, ICIIS'99, 1999.

[2] M. Calder, V. Vyshemirsky, D. Gilbert, R. Orton, Analysis of signaling pathways using the prism model checker, in: Proc. Computational Methods in Systems Biology, CMSB'05, 2005, pp. 179–190.

[3] L. Calzone, F. Fages, S. Soliman, Biocham: An environment for modeling biological systems and formalizing experimental knowledge, Bioinformatics 22 (14) (2006) 1805–1807.

[4] S. Efroni, D. Harel, I.R. Cohen, Toward rigorous comprehension of biological complexity: Modeling, execution, and visualization of thymic t-cell maturation, Genom. Res. 13 (11) (2003) 2485–2497.

[5] D.D. Errampalli, C. Priami, P. Quaglia, A formal language for computational systems biology, Omics 8 (4) (2004) 370–380.

[6] J. Fisher, N. Piterman, E.J. Hubbard, M.J. Stern, D. Harel, Computational insights into caenorhabditis elegans vulval development, Proc. Natl. Acad. Sci. USA 102 (6) (2005) 1951–1956.

[7] D. Harel, S. Efroni, I.R. Cohen, Reactive animation, Lect. Notes Comput. Sci. 2852 (2003) 136–153.

[8] N. Kam, I.R. Cohen, D. Harel, The immune system as a reactive system: Modeling t cell activation with statecharts, in: Proc. Visual Languages and Formal Methods, VLFM'01, part of IEEE Symp. on Human-Centric Computing, HCC'01, 2001, pp. 15–22.

[9] N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, E.J. Hubbard, M.J. Stern, Formal modeling of c. Elegans development: A scenario-based approach, in: Computational Methods in Systems Biology: First International Workshop, 2602 / 2003, 2003, pp. 4–20.

[10] A. Regev, W. Silverman, E. Shapiro, Representation and simulation of biochemical processes using the pi-calculus process algebra, Pac. Symp. Biocomput. (2001) 459–470.

[11] A. Sadot, J. Fisher, D. Barak, Y. Admanit, M.J. Stern, E.J. Hubbard, D. Harel, Towards verified biological models, IEEE/ACM Trans. Comput. Biology and Bioinformatics (in press).

[12] S.Y. Shvartsman, C.B. Muratov, D.A. Lauffenburger, Modeling and computational analysis of egf receptor-mediated cell communication in drosophila oogenesis, Development 129 (11) (2002) 2577–2589.

[13] C. Talcott, D.L. Dill, The pathway logic assistant, in: Proc. Computational Methods in Systems Biology, CMSB'05, 2005, pp. 228–239.

[14] S. Efroni, D. Harel, I.R. Cohen, Emergent dynamics of thymocyte development and lineage determination, PLoS Comput. Biol. 3 (1) (2007) e13.

[15] M. Meier-Schellersheim, X. Xu, B. Angermann, E.J. Kunkel, T. Jin, R.N. Germain, Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method, PLoS Comput. Biol. 2 (7) (2006) e82.

[16] B.M. Slepchenko, J.C. Schaff, I. Macara, L.M. Loew, Quantitative cell biology with the virtual cell, Trends Cell. Biol. 13 (11) (2003) 570–576.
[17] A. Csikasz-Nagy, D. Battogtokh, K.C. Chen, B. Novak, J.J. Tyson, Analysis of a generic model of eukaryotic cell-cycle regulation, Biophys. J. 90 (12) (2006) 4361–4379.
[18] R. Segev, E. Ben-Jacob, Generic modeling of chemotactic based self-wiring of neural networks, Neural Netw. 13 (2) (2000) 185–199.
[19] International Human Genome Sequencing Consortium, Finishing the euchromatic sequence of the human genome, Nature 431 (7011) (2004) 931–945.
[20] I.R. Cohen, H. Atlan, Genetics as explanation: Limits to the human genome project, Encyclopedia Life Sci. (2006).
[21] I.R. Cohen, Tending Adam's Grden, Academic Press, San Diego, CA, 1999.
[22] D. Harel, Statecharts: A visual formalism for complex systems, Sci. Comput. Programm. 8 (1987) 231–274.
[23] D. Harel, E. Gery, Executable object modeling with statecharts, Computer 30 (7) (1997) 31–42.
[24] D. Drusinsky, Model checking of statecharts using automatic white box test generation, circuites and systems, in: 48th Midwest Symposium on, 2005, pp. 327–332.
[25] S. Efroni, D. Harel, I.R. Cohen, Reactive animation: Realistic modeling of complex dynamic systems, Computer 38 (1) (2005) 38–47.
[26] I.R. Cohen, D. Harel, Explaining a complex living system: Dynamics, multi-scaling and emergence, J. Royal Soci. Inter. 4 (2007) 175–182.