Formal Modeling of C. elegans Development:
A Scenario-Based Approach

Na'aman Kam[1], David Harel[1], Hillel Kugler[1], Rami Marelly[1], Amir Pnueli[1],
E. Jane Albert Hubbard[2], and Michael J. Stern[3]

[1] Dept. of Computer Science and Applied Mathematics, The Weizmann Institute of
Science, Rehovot 76100, Israel
{kam,dharel,kugler,rami,amir}@wisdom.weizmann.ac.il
[2] Dept. of Biology, New York University, New York, NY
jane.hubbard@nyu.edu
[3] Dept. of Genetics, Yale University School of Medicine, New Haven, CT
michael.stern@yale.edu
http://www.wisdom.weizmann.ac.il/~kam/
CelegansModel/CelegansModel.htm

Abstract. We present preliminary results of a new approach to the formal modeling of biological phenomena. The approach stems from the conceptual compatibility of the methods and logic of data collection and analysis in the field of developmental genetics with the languages, methods and tools of scenario-based reactive system design. In particular, we use the recently developed methodology consisting of the language of *live sequence charts* with the *play-in/play-out* process, to model the well-characterized process of cell fate acquisition during *C. elegans* vulval development.

1 Introduction

Our understanding of biology has become sufficiently complex that it is increasingly difficult to integrate all the relevant facts using abstract reasoning alone. This is exacerbated by current high throughput technologies that spew data at ever-increasing rates. While bioinformatic approaches to handling this mass of data have generated databases that ease the storage and accessibility of the data, rigorous modeling approaches are necessary to integrate these data into useable models that can exploit and analyze the available information. There are many current efforts aimed at biological modeling, and it is likely that different approaches will be appropriate for various types of biological information and for various research objectives [1, 2]. Here, we present a novel approach to modeling biological phenomena. It utilizes in a direct and powerful way the mechanisms by which raw biological data are amassed, and smoothly captures that data within tools designed by computer scientists for the design and analysis of complex reactive systems.

A considerable quantity of biological data is collected and reported in a form that can be called "condition-result" data. The gathering is usually carried out by initializing an experiment that is triggered by a certain set of circumstances (conditions), following which an observation is made and the results recorded. The condition is most often a perturbation, such as mutating genes or exposing cells to an altered environment. For example, genetic data often first emerge as phenotypic assessments (anatomical or behavioral outputs) that are compared between a mutant background and a defined "wild-type." Another example includes observations of the effects of anatomical manipulations (e.g. cell destruction or tissue transplantation) on the behavior of the remaining structures. These types of experiments test specific hypotheses about the nature of the system that is perturbed. Many inferences about how biological systems function have been made from such experimental results, and our consequent understanding based on these logical inferences is becoming increasingly, even profoundly, complex.

One feature of these types of experiments is that they do not necessitate an understanding of the particular molecular mechanisms underlying the events. For example, much information can be ascertained about a gene's function by observing the consequences of loss of that function before the biochemical nature or activity of the gene product is known. Moreover, even when the biochemical activity is known, the functional significance of that activity in the context of a biological system is often deduced at the level of phenotypic output. Naturally, with knowledge of molecular mechanisms, increasingly sophisticated inferences can be made and more detailed hypotheses tested, but the outputs, be they a certain cell fate acquisition, changes in gene expression patterns, etc., are often recorded and analyzed at the level of a phenotypic result. Thus, a large proportion of biological data is reported as stories, or "scenarios," that document the results of experiments conducted under specific conditions. The challenge of modeling these aspects of biology is to be able to translate such "condition-result" phenomena from the "scenario"-based natural language format into a meaningful and rigorous mathematical language. Such a translation process will allow these data to be integrated more comprehensively by the application of high-level computer-assisted analysis. In order for it to be useful, the model must be rigorous and formal, and thus amenable to verification and testing.

We have found that modeling methodologies originating in computer science and software engineering, and created for the purpose of designing complex *reactive systems*, are conceptually well suited to model this type of condition-result biological data. Reactive systems are those whose complexity stems not necessarily from complicated computation but from complicated reactivity over time. They are most often highly concurrent and time-intensive, and exhibit hybrid behavior that is predominantly discrete in nature but has continuous aspects as well. The structure of a reactive system consists of many interacting components, in which control of the behavior of the system is highly distributed amongst the components. Very often the structure itself is dynamic, with its components being repeatedly created and destroyed during the system's life span.

The most widely used frameworks for developing models of such systems feature *visual formalisms*, which are both graphically intuitive and mathematically rigorous. These are supported by powerful tools that enable full model executability and analy-

sis, and are linkable to graphical user interfaces (GUIs) of the system. This enables realistic simulation prior to actual implementation. At present, such languages and tools --- often based on the *object-oriented* paradigm --- are being strengthened by verification modules, making it possible not only to execute and simulate the system models (test and observe) but also to verify dynamic properties thereof (prove).
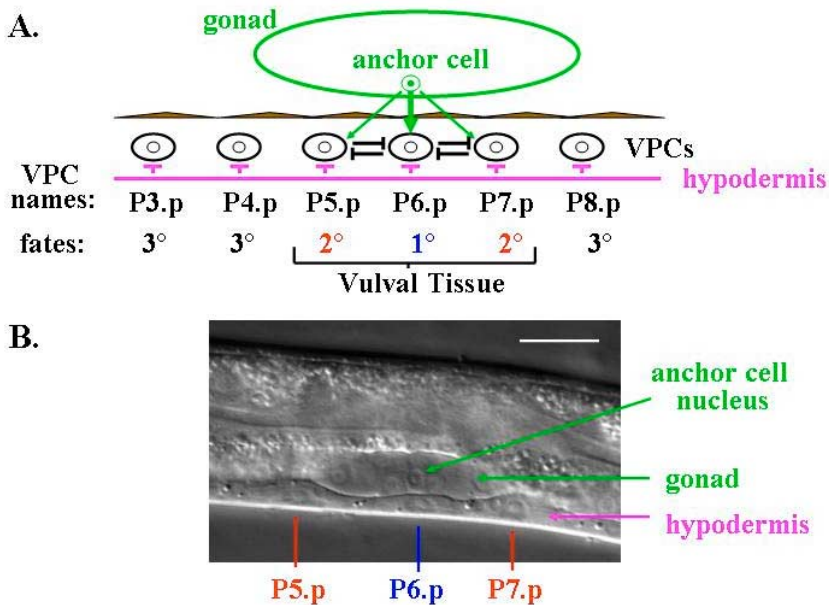

Fig. 1. Vulval cell fate determination. (A) Schematic representation of cellular interactions that determine vulval fates. Signals (arrows and T-bars) are color-coded based on their source. The anchor cell promotes vulval fates. VPC-VPC interactions inhibit adjacent cells from acquiring a 1° fate (and instead promote a 2° fate). The ventral hypodermis inhibits vulval (1° and 2°) fates. (B) Differential interference contrast (DIC) photomicrograph of the central body region of a live *C. elegans* L3-stage worm. White bar ~ 15μm

A central premise of this paper is that many kinds of biological systems exhibit characteristics that are remarkably similar to those of reactive systems. The similarities apply to many different levels of biological analysis, including those dealing with molecular, cellular, organ-based, whole organism, or even population biology phenomena. Once viewed in this light, the dramatic concurrency of events, the chain-reactions, the time-dependent patterns, and the event-driven discrete nature of their behaviors, are readily apparent. Consequently, we believe that biological systems can be productively modeled as reactive systems, using languages and tools developed for the construction of man-made systems.

In previous work on T cell activation [3] and T cell behavior in the thymus [4], the feasibility of this thesis was addressed on a small scale, and the results have been very encouraging. In particular, that work demonstrated the adequacy of object-oriented analysis to modeling biological systems, and showed the applicability of the visual formalism of statecharts for representing their behavior.

In our current work, we not only begin to tackle a more complex system, but also incorporate two levels of data into our model. One of these levels, shared also by the models from the previous work, represents the 'rules' of the behavior of the system. These rules are based on abstractions and inferences from various sets of primary data. The second level of data being incorporated into our model, is new, and it includes the "observations" that comprise the primary data itself. The set of observations utilized is a crucial component of the model, allowing both execution and verification, including consistency checks between the observations and the inferred rules. To accomplish this, we also depart in a significant way from the intra-object, state-based, statechart approach used in the previous work, and instead use a more recently-developed inter-object, scenario-based approach to reactive system specification. The language we use to formalize the data is called *live sequence charts* (LSCs) [5], and capturing the data and analyzing it is carried out using the *play-in/play-out* methodology, supported by the *Play-Engine* tool [6,7].

1.1   The Biological System

Our current effort is focused on a means to formalize and analyze primary data such that the consistency of inferences made from these data can be tested as part of the model. We have chosen the development of the nematode *Caenorhabditis elegans* as a subject since this organism is extremely well defined at the anatomical, genetic, and molecular level. Specifically, the entire cell lineage of *C. elegans* has been traced, many genetic mutants have been described, and the entire genome is sequenced [8, 9]. Moreover, virtually all of the researchers working on this organism use conceptually similar methodologies and logic in their genetic experiments, and use a uniform substrate wild-type strain (N2). These circumstances make feasible computer-assisted integration of the primary data. Finally, genome-wide efforts to define the function, expression levels, expression patterns and interactions of all genes are underway [10–13].

As a specific test-case, we have begun to model *C. elegans* vulval development (see [14] for a recent review). The vulva is a structure through which eggs are laid. This structure derives from three cells within a set of six cells with an equivalent set of multiple developmental potentials (Fig. 1). These six cells are named P3.p, P4.p, P5.p, P6.p, P7.p and P8.p (collectively termed P(3-8).p). Due to their potential to participate in the formation of the vulva, they are also known as the vulval precursor cells (VPCs). Each has the potential to acquire either a non-vulval fate (a 3º fate) or one of two vulval cell fates (a 1º or a 2º cell fate; see Fig. 1). During normal development, after a series of cell divisions in a characteristic pattern, a vulva consisting of 22 nuclei is formed. Vulval development was one of the first areas to which considerable effort was applied to achieve a molecular understanding of how cells acquire their particular fates. This system, though limited in cell number, is quite complex and ultimately integrates at least four different molecular signaling pathways in three different interacting tissue types. Cell fate acquisition in development also occurs in the context of cell cycle control and the general global controls on the growth and development of the organism. Hence, vulval development indeed represents a rather complex reactive system.

Here, we describe our progress applying the visual formalism of LSCs and the play-in/play-out methodology to the representation of biological scenarios from one particular study, the data published in a paper by Sternberg and Horvitz in 1986 [15]. LSCs appear to be highly accessible to biologists, while retaining mathematical rigor. Designed specifically to capture scenario-based behavior, the structure of LSCs fits extremely well into our framework of condition-result data. In this paper, data are reported regarding a series of experiments in which cells are destroyed using a laser microbeam (ablation). Since signals among the VPCs and between the VPCs and adjacent tissues cooperate to determine the fates of these cells, these experiments test the nature of these interactions and the relative potential of the VPCs to adopt vulval (versus non-vulval) fates and, among those that adopt a vulval fate, the specific type of vulval fate. These data are used to infer specific properties of the events that occur in the wild-type situation and to generate a "model" for how the unperturbed system works based on the behavior of the perturbed system. We present a subset of this test-case below with special emphasis on the LSC language and the play-in/play-out methodology. In addition, we present our solutions to particular challenges posed by the nature of biological data itself and its formalization from the natural language used by *C. elegans* biologists. In representing even a small and simple set of experimental results, we have addressed a number of issues that provide evidence for the flexibility and potential of this modeling approach.

1.2   The Modeling Methodology

We are adopting an inter-object, scenario-based modeling approach, using the language of *live sequence charts* (LSCs) [5] and the *play-in/play-out* methodology [6, 7], both supported by the *Play-Engine* modeling tool [6,7]. The decision to take this approach, rather than the statechart-based one, emerged from the consideration of how to best represent the C. elegans data formally, and how to best carry out the formalization process.

LSCs constitute a visual formalism for specifying sequences of events and message passing activity between objects. They can specify scenarios of behavior that cut across object boundaries and exhibit a variety of modalities, such as scenarios that can occur, ones that must occur, ones that may not occur (called *anti-scenarios*), ones that must follow others, ones that overlap with others, and more. Technically, there are two types of LSCs, universal and existential. The elements of LSCs (events, messages, guarding conditions, etc.) can be either mandatory (called *hot* in LSC terminology) or provisional (called *cold*). Universal charts are the more important ones for modeling, and comprise a *prechart* and a *main* chart, the former triggering the execution of the latter. Thus, a universal LSC states that whenever the scenario in the prechart occurs (e.g., the user has flipped a switch), the scenario in the main chart must follow it (e.g., the light goes on). Thus, the relation between the prechart and the chart body can be viewed as a dynamic condition-result: if and when the former occurs, the system is obligated to satisfy the latter.

Play-in/play-out is a recently developed process for modeling in LSCs, with which one can conveniently capture inter-object scenario-based behavior, execute it, and simulate the modeled system in full. The play-in part of the method enables people

who are unfamiliar with LSCs to specify system behavior using a high level, intuitive and user-friendly mechanism. The process asks that the user first build the graphical user interface (GUI) of the system, with no behavior built into it. The user then 'plays' the GUI by clicking the graphical control elements (in electronic systems these might be buttons, knobs, and so on), in an intuitive manner, in this way giving the engine sequences of events and actions, and teaching it how the system should respond to them. As this is being done, the Play-Engine continuously constructs the corresponding LSCs automatically.

While play-in is the analogue of writing programs, play-out is the analogue of running them. Here the user simply plays the GUI as he/she would have done when executing the real system, also by clicking buttons and rotating knobs, and so on, but limiting him/herself to end-user and external environment actions. As this is going on, the Play-Engine interacts with the GUI and uses it to reflect the system state at any given moment. The scenarios played in using any number of LSCs are all taken into account during play-out, so that the user gets the full effect of the system with all its modeled behaviors operating correctly in tandem. All specified ramifications entailed by an occurring event or action will immediately be carried out by the engine automatically, regardless of where in the LSCs it was originally specified. Also, any violations of constraints (e.g., playing out an anti-scenario) or contradictions between scenarios, will be detected if attempted. This kind of sweeping integration of the specified condition-result style behavior is most fitting for biological systems, where it can often be very difficult to connect the many pieces of behavioral data that are continuously discovered or refined.

2   Results

2.1   The GUI of Vulval Fate Determination

A critical aspect of the GUI is that it can be designed by the user to reflect his/her view of the system. In our case, the GUI (Fig. 2) is a simplified representation of the actual anatomical situation under study (Fig. 1). The six VPCs and the interacting adjacent tissues, that include the gonad and the ventral hypodermis, are represented. This GUI represents the developmental decisions that occur during a discrete window of time, and we developed it the way we did in order for it to best capture that particular stage in the organism's development It can be expanded to capture more parts and stages in the development process, and also to include inputs from other GUIs (possibly constructed by other people working on a growing distributed model) that represent other connected developmental vignettes. The beauty of this kind of GUI is that it can be made to directly reflect the way biologists represent their system's anatomy, and the particular GUI in Figure 2 is intuitive for anyone working on vulval development. However, the more important aspect of the GUI is that it allows the model to "come alive" in the context of the play-in/play-out approach.
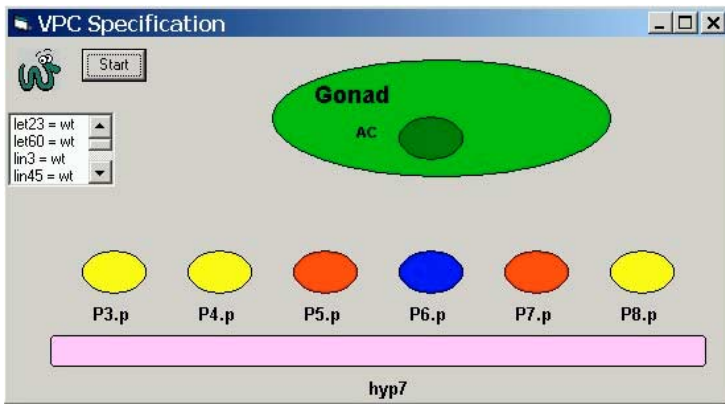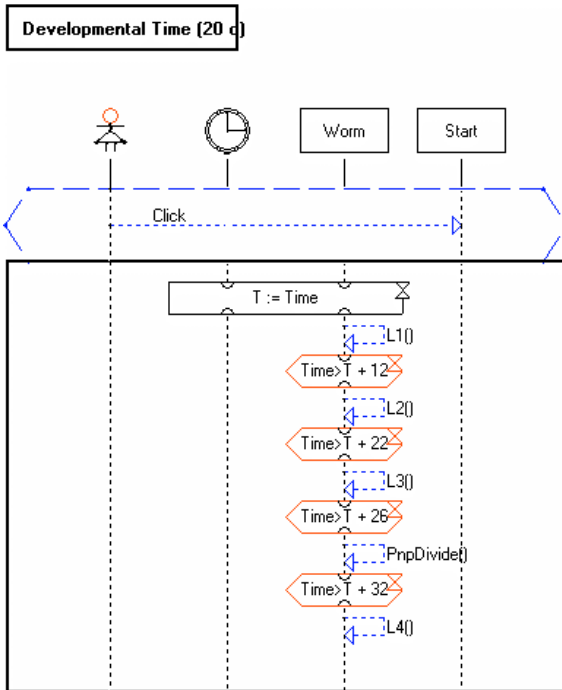
Fig. 2. The GUI

Fig. 3. An LSC depicting developmental time

2.2   Case Study: Sternberg and Horvitz, 1986

We present several examples of scenarios that were described in [15], which we translated into LSCs. These examples demonstrate how data is entered into the model and give context to the progress we have made regarding broad conceptual aspects of

the formalization of biological data. Concepts already incorporated into the model include the representation of developmental time, symbolic instances, default assumptions, and non-deterministic behavior. Examples of LSCs are included that represent the formalization of both primary observations and inferred rules.

Developmental Time. In the current model, developmental time drives the behavior of the whole system. (See [16] for the way the issue of time is treated in LSCs and the Play-Engine.) Thus, we begin by presenting the corresponding LSC (Fig. 3). We decided to concentrate on post-embryonic development (after the embryo has hatched from the egg), but earlier developmental stages can easily be incorporated as well. Post-embryonic development is divided into four larval stages (denoted L1 to L4), each of which end in a molt. The time units represent hours from hatching (the beginning of L1). This LSC is activated when the user clicks the start button on the GUI. The time at which this operation was performed is stored in a variable called T, and all other events in this LSC refer to this time point. Thus, for example, the L1 phase lasts 12 hours, the L2 phase lasts 10 hours, and the Pn.p cells divide 26 hours after hatching, which is 4 hours after entry into L3.
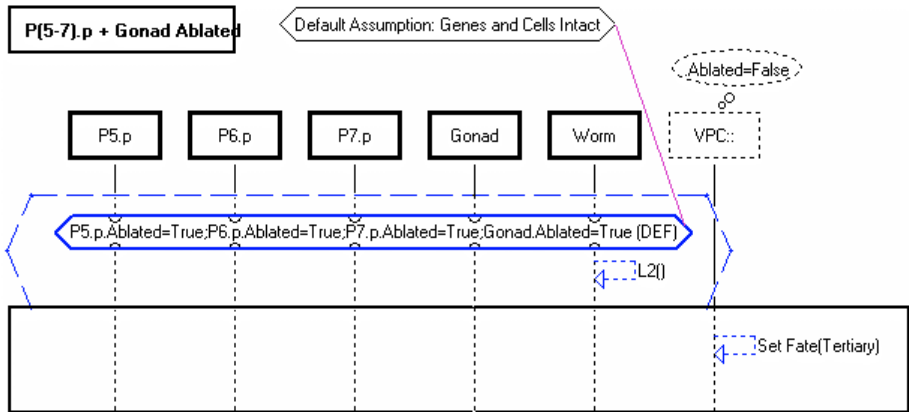

Fig. 4. A scenario involving the ablation of P(5-7).p and the gonad

Representing the Experimental Setup in the Prechart. The experiment represented by the LSC in Fig. 4 was devised to establish the ground-state fate of the P(3,4,8).p cells, that is to see what cell fate these cells would acquire if they were not influenced by any known interactions from adjacent cells. At the time of this publication, the influence of the hypodermis on vulval fate specification was not known (it is also irrelevant under these specific conditions). The cells known to influence VPC cell fate were removed by laser ablation, including the three VPCs that are normally induced to form the vulva (P5.p, P6.p and P7.p). The fates of the remaining VPCs were observed and recorded. The results of this set of experiments were reported in the following statement:
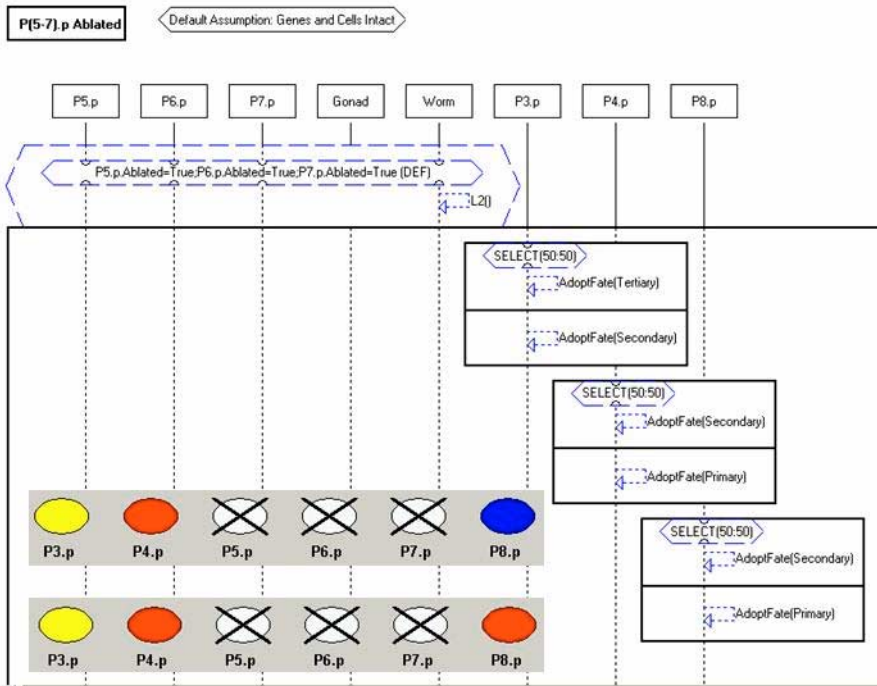
Fig. 5. Representing non-deterministic events. The results of two consecutive executions of this LSC are shown. It can be seen that P8.p adopted a different fate on each of these executions

> *"We ablated P(5-7).p and the gonad in five L1 hermaphrodites; in these animals P(3,4,8).p each adopted a tertiary fate"* [15].

In this experiment, the cell ablations took place during the first larval stage (L1). This fact is stated in the prechart by placing the condition before the L2 event: the prechart will be satisfied only if by the time the L2 event occurs (determined in the developmental time LSC) the specified cells will already be ablated.

Experimental setups, such as the above one, implicitly assume that besides the reported perturbations all other elements of the system remained intact. We represent such implicit assumptions in a set of *default assumptions*, which includes all the properties of the system for which we would like to set some default value (e.g., genes are not mutated, or "wild type"). If a given condition within the LSC includes assigning a non-default value to some property, this requirement will override the default assumption. Thus, the condition that appears in the pre-chart of Fig. 4 corresponds to a situation in which P(5-7).p and the gonad were ablated *while all other genes and cells remained intact*. During a play-in session, a condition can be bound to a set of default assumptions by clicking a checkbox within the condition window. Within the LSC, this action is reflected by the DEF (default) tag that appears at the end of the condition, as well as by a line that connects the condition with the set of default assumptions that appears at the top of the LSC (the line appears only when the mouse cursor is placed over the condition).

Symbolic Instances. The LSC in Fig. 4 illustrates the concept of symbolic instances. (See [17] for how symbolic instances are dealt with in LSCs and the Play-Engine.) In this experiment, the behavior of all remaining VPCs is similar: they all adopt tertiary fates. This is expressed in the LSC by making a general statement that applies to a symbolic instance of the VPC class. This depiction indicates that during execution, all instances of the VPC class that were not ablated will be bound to this LSC and execute a tertiary fate.

Non-deterministic Behavior. Part of the experiment designed to assess the ground state of the VPCs involved ablating the cells that normally adopt vulval fates [P(5-7).p] but leaving the gonad intact. Three animals were observed after performing these ablation conditions; the fates of the VPCs are shown below (reported in Table 3, line 3, [15]):

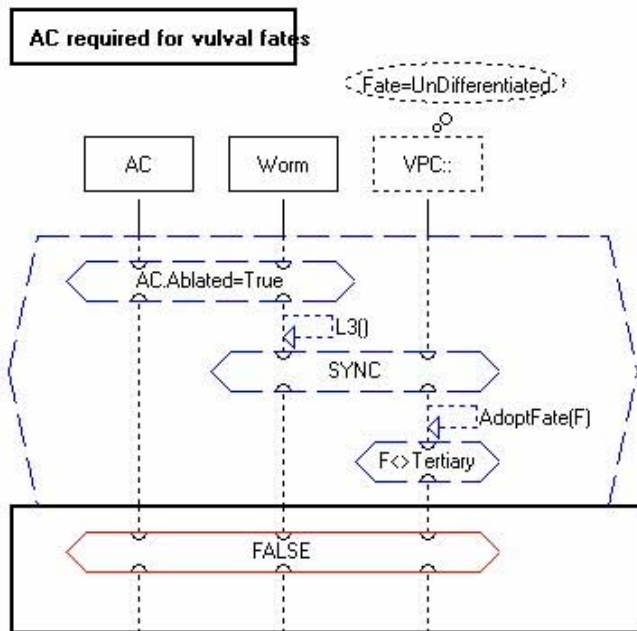| Pn.p cells:	P3.p	P.4p	P5.p	P6.p	P7.p	P8.p
Fates:	2° or 3°	1° or 2°	ablated	ablated	ablated	1° or 2°

Fig. 6. An anti-scenario

As is often the case in biological experiments, the results of the conditions of this experiment were not deterministic: P3.p adopted either a 2° or a 3° fate, while P4.p and P8.p adopted either a 1° or a 2° fate. Each of these fates is recognizable by a specific pattern of cell divisions and cellular morphologies. Non-determinism is represented in the LSC that depicts this experiment (Fig. 5) using selection boxes within the

main chart. This representation enables the model to execute non-deterministic be-
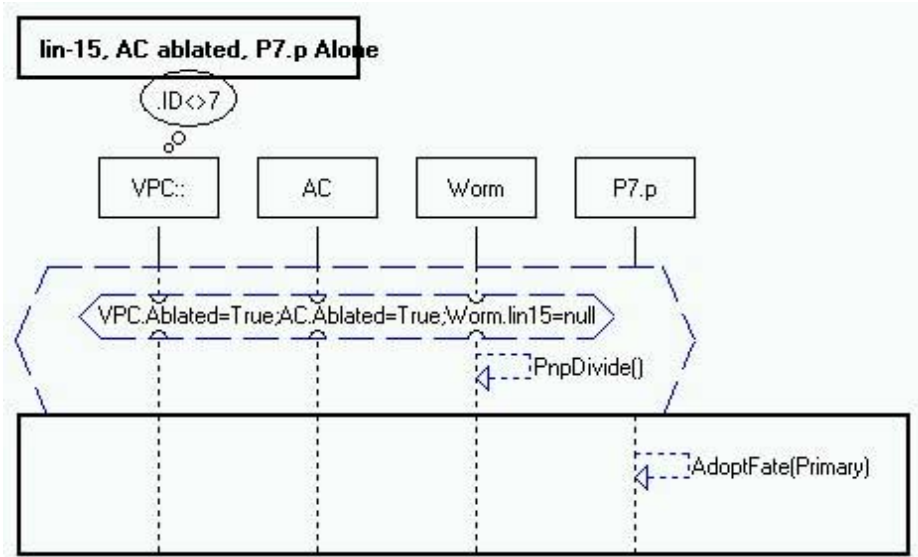havior of objects using a selected list of outcomes at prescribed frequencies.

Fig. 7. A VPC can adopt a vulval fate in the absence of the anchor cell

Representing General Rules as Anti-scenarios. In addition to the input of raw data
from specific experiments, general rules of system behavior can also be included as
LSCs. Some rules can be represented most easily by what are known as *anti-
scenarios*. For example, to represent the statement:

> *"The anchor cell is required for Pn.p cells to adopt vulval (1° or 2°) as op-
> posed to non-vulval (3°) fates" [15],*

an LSC representing the following anti-scenario can be used:

"it cannot be the case that a VPC will adopt a vulval fate in the absence of the anchor
cell" (Fig. 6).

The ability to represent behavior using "must" and "must not" statements lends greater
potential and power to modeling efforts.

System Analysis: The Play-Out Mechanism. The play-out mechanism can be used
to test the system in various ways:

1. *Detecting inconsistencies among LSCs.* In its anti-scenario guise, an LSC
   representing, for example, a scenario in which a VPC adopts a vulval fate in
   the absence of the anchor cell would cause the Play-Engine to announce that
   a 'hot' condition was violated. Certain mutations can obviate the requirement
   of the anchor cell for vulval cell fates. For example, if the hypodermal in-
   hibitory mechanism (see Fig. 1) is compromised by mutation, vulval fates can

occur even in the absence of the gonad (e.g. see [19]). Executing a play-out session in which an 'experiment' of this type was performed resulted in a violation of the 'old' rule. Thus, this modeling approach can help integrate new results into the framework of existing data, pointing out inconsistencies that might have been ignored by the experimentalist.

2. *Predictability*. The play-engine can juxtapose LSCs generated either from specific information results or from general biological rules, which are inferred from many data sets. This juxtaposition has the potential to simulate novel scenarios and to highlight behaviors that were not previously observed. One such example was observed over our model while playing-out a scenario that involved a combination of two mutations ('double mutant'), one in the dig-1 gene and the other in the lin-12 gene. In dig-1 mutants the gonad is shifted anteriorly, and in lin-12(0) mutants the three VPCs P(5-7).p were observed to adopt a 1° fate. An actual experiment that detects the effect of combining these two mutations together has not been done yet. However, based on the LSCs that represent the observed phenotypes for each individual mutant, together with a couple of LSC that represent deduced rules regarding signaling mechanisms involved in vulval induction, the play-engine predicted that such an experiment would result in P(4-6).p adopting a 1° fate. Such a play-out session illustrates that the play-engine can execute scenarios that were not played-in explicitly. (For further explanations, including a demo of this execution, see [19].)

3. *Query the system for scenarios that satisfy a given behavior*. Play-out has been extended with a powerful new module, called *smart play-out* [18], which utilizes tools from program verification in order to analyze an LSC model in much richer ways than just executing it. Many of its uses come from asking the system to find a way to do something on its own, or to ask it "Is this possible?" kinds of questions. For example, we can ask the system to automatically figure out if there is some possible way of satisfying a particular scenario, and to then run the resulting discivered sequence on its own. An example for such a test is depicted in Figure 7, in which an existential LSC is used to query the system for scenarios that result in P7.p adopting a 3° fate. (For further explanations, including a demo of applying this test to the model, see [19].)

Thus, play-out, with its smart play-out enrichment, makes it possible to detect, or predict, the outcome of combinations of conditions not previously tested, and to query the system for many different kinds of desired (or undesired) outcomes.
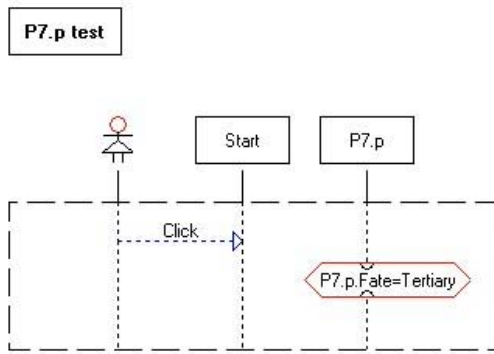
Fig. 8. An existential chart is used to test whether there is any scenario in which P7.p adopts a primary fate

3   Discussion

3.1   Strengths of Our Approach

In previous sections, we illustrated the capability of our modeling approach to facilitate the transformation of biological data into formal computer-analyzable statements. In particular, our modeling approach provides solutions for the following challenges involved in this formalization process:

Incorporating Raw Biological Data. One of the unique features of our approach is the incorporation of primary data, rather than modeling the general rules that are derived or inferred from the actual data. One challenge to modeling primary data is the integration of information obtained in different ways and using different approaches. The assessment of results can vary with the individual experimentalist and changes over time as new information about the system and technologies to evaluate it become available. The variety of data collection methods can be integrated by incorporating LSCs that link the different types of data together. We can then use the model to validate equivalence statements against the data, thus enhancing our understanding of the biological system.

Importantly, primary data, once entered as LSCs, can be used even when the general rules of behavior are modified with time. Primary data scenarios can be derived from many different laboratories and many different types of experimentation, and can be used to detect conflicts within the primary data, highlighting important aspects to be later tested more rigorously.

Rules as a Second Level. A computer-assisted approach to a formalized list of primary data is useless unless it can be tested for overall consistency and logical conflict, in ways that are better than can be done by human reasoning alone. This possibility is also built into our approach. A logical system is routinely used by experimentalists to infer abstractions about the real system from condition-result types of data. In our

model, these abstractions can be formalized as 'rules'. For example, they can be entered in the form of anti-scenarios (as shown above). Thus, it may be possible to formalize a system that is already in use in a non-formal way so that the rigorous tools of computer-based verification can be brought to bear on the data. Moreover, having both primary data and associated 'rules', allows novel scenarios to be obtained from the input data, as illustrated above.

Non-deterministic Events. One of the hallmarks of biological data is non-determinism. While some non-determinism likely derives from the paucity of our knowledge of the important details of the systems of interest, the very nature of biological systems contains some inherent non-determinism. For example, stochastic events, threshold phenomena and feedback systems, all operate at the molecular and cellular levels and it is crucial that methods and tools for their modeling support explicit choice-based, or probability-based, branching of behavior.

3.2   What Can We Get Out of It?

A Behavioral Database. At the most basic level, a model of the type we propose forms a behavioral database that can be used for retrieving dynamic data. Existing biological databases can be used to retrieve the sequence of a given gene, or even to find its homologues, but there is no current mechanism for asking questions such as 'what will happen if gene X will be mutated together with gene Y', or 'what will be the phenotypic outcome of ablating cell A over a specific genetic background'.

Detecting Inconsistencies. New inferences that are made can be checked for consistency against existing observations and rules. As illustrated in section 2.2 above, 'old' rules that are contradicted by new experiments can be detected as well.

Predictions. The play-out approach offers the additional possibility of asking the system for the predicted result of an experiment that has not yet been performed. Such predictions can then be tested in the lab. If the model predicts an outcome different from the actual outcome, there are likely factors that have not been included in the model that are relevant to the experimental system. Identifying such putative factors can then lead to new experiments that will aim at filling in these gaps. Alternatively, such a mismatch between prediction and experiment can result from improper rules or interpretations that currently constitute the model. Thus, the kind of model analysis we propose can assist in improving, correcting and sharpening our understanding of the biological system.

Explaining Surprising Results. The smart play-out mechanism can be used to answer questions such as: 'Is there any scenario in which a given behavior will be observed?' Such tests can be applied to explaining surprising experimental outcomes that were observed in the lab ('based on the current data, is there any scenario that can produce this surprising result?'), or to test our understanding of the system ('is there indeed no scenario, in which, although the product of gene X was eliminated, a given tissue T will still develop normally?').

3.3   Current Status and Future Directions

To date, we have formalized as LSCs only a small set of data pertaining to VPC speci-fication. These have served to highlight some of the critical issues that need to be addressed in modeling this system in its entirety. Several of these issues have already been addressed, as described in this paper. In addition to the conceptual advances this small data set has prompted, these LSCs reveal much of the promise and feasibility of applying LSCs and play-in/play-out to the modeling of biology. One of the strengths of the entire approach is the ability to execute a model even with incomplete data sets. As we continue to fill out our test-case model, including the integration of different signal-transduction inputs into the determination of vulval fates, we are also consid-ering the expansion of the methodology in several areas:

Expanding the Experimental Repertoire. Although our current efforts are concen-trated on condition-result data from genetic and anatomical manipulations, there is no *a priori* reason why the same methods should not be applied to other types of data, representing other levels of biological inquiry. These include biochemical data (such as signal transduction pathways, protein-protein interactions, etc.) and gene expression data (microarray data, anatomical expression pattern information, etc.).

Distributed Play-In/Play-Out. It is already possible to connect separate GUIs and Play-Engines to each other. This will enable multiple labs to participate in the model-ing effort. Each lab can design its own GUI to represent its sub-system of interest, and then play in the relevant scenarios. Connecting Play-Engines to each other can facili-tate a distributed play-out mechanism, in which an event that occurs in an LSC that is being executed on one computer will activate and LSC that belongs to a specification running on another computer. The Play-Engine development team is also working on tolls to connect the Play-Engine to other environments as well, e.g., tools that enable statechart-based modeling, such as Rhapsody, and ones that deal with the continuous aspects of systems. These too will significantly expand the possibilities of modeling biological systems.

   Another possibility is to connect to the C. elegans field's database Wormbase (or its equivalent in other systems), so that just as we can obtain the sequence and homo-logues, etc., for each gene, we would be able to retrieve via Wormbase the LSCs in which it participates.

The Long-Term Goal. Finally, we truly believe that this research effort could turn out being the first step in a far, far more ambitious project. Namely, to construct a full 4-dimensional model of a multi-cellular animal, which is true to all known facts about it, and which is easily extendable as new facts are discovered (see [21] for a more detailed discussion of this).

References

[1]   Bower, J.M., Bolouri, H. (eds.): Computational modeling of genetic and bio-chemical networks. The MIT Press, Cambridge, MA (2001)

[2]   Wilkins, A.S. (ed.): Modelling complex biological systems: a special issue. BioEssays 24(12) Wiley Periodicals, Inc., Hoboken, NJ (2002)

[3]   Kam, N., Cohen, I.R., Harel, D., The Immune System as a Reactive System: Modeling T Cell Activation with Statecharts. To appear in Bull. Math. Bio. An extended abstract of this paper appeared in Proc. Visual Languages and Formal Methods (VLFM01), part of IEEE Symposia on Human-Centric Computing Languages and Environments (HCC01), pp. 15-22 (2001)

[4]   Efroni, S., Harel, D. and Cohen I.R., Reactive Animation, submitted (2002)

[5]   Damm, W. and Harel, D., LSCs: Breathing Life into Message Sequence Charts, Formal Methods in System Design 19:1 (2001). (Preliminary version in Proc. 3rd IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'99), (P. Ciancarini, A. Fantechi and R. Gorrieri, eds.), Kluwer Academic Publishers, 1999, pp. 293-312.)

[6]   Harel, D. and Marelly, R Come, Let's Play: A Scenario-Based Approach to Programming, Springer- Verlag, to appear (2003)

[7]   Harel, D. and Marelly, R., Specifying and Executing Behavioral Requirements: The Play In/Play-Out Approach, Software and System Modeling (SoSyM), to appear (2003)

[8]   Riddle, D.L., Blumenthal, T. , Meyer, B.J., Priess, J.R. (eds.): C. elegans II Cold Spring Harbor Laboratory Press Plainview, NY (1997)

[9]   The C. elegans Sequencing Consortium: Genome sequence of the nematode C. elegans: a platform for investigating biology. The C. elegans Sequencing Consortium Science 282 (1998) 2012-2018

[10]   Fraser, A.G., Kamath, R.S., ZipperlenI, P., Martinez-Campos, M., Sohrmann, M., Ahringer, J.: Functional genomic analysis fo C. elegans chromosome I by systematic RNA interference Nature 408 (2000) 325-330

[11]   Piano, F., Schetter, A. J., Mangone, M., Stein, L., Kemphues, K. J.: RNAi analysis of genes expressed in the ovary of Caenorhabditis elegans Curr Biol 10(24) 2000 1619-1622

[12]   Gonczy, P., Echeverri, C., Oegema, K., Coulson, A., Jones, S. J., Copley, R. R., Duperon, J., Oegema, J., Brehm, M., Cassin, E., Hannak, E., Kirkham, M., Pichler, S., Flohrs, K., Goessen, A., Leidel, S., Alleaume, A. M., Martin, C., Ozlu, N., Bork, P., Hyman, A. A.: Functional genomic analysis of cell division in C. elegans using RNAi of genes on chromosome III Nature 408 (2000) 331-336

[13]   Maeda, I., Kohara, Y., Yamamoto, M., Sugimoto, A.: Large-scale analysis of gene function in Caenorhabditis elegans by high-throughput RNAi Curr Biol 11(3) (2000) 171-176

[14]   Wang, M., Sternberg, P. W.: Pattern formation during C. elegans vulval induction Curr Top Dev Biol 51 (2001) 189-220

[15]   Sternberg, P. W., Horvitz, H. R.: Pattern formation during vulval development in C. elegans Cell 44 (1986) 761-772

[16] Harel, D. and Marelly, R., Playing with Time: On the Specification and Execu-
      tion of Time-Enriched LSCs, Proc. 10th IEEE/ACM Int. Symp. on Modeling,
      Analysis and Simulation of Computer and Telecommunication Systems
      (MASCOTS 2002), Fort Worth, Texas (2002)
[17] Marelly, R., Harel, D. and Kugler, H., Multiple Instances and Symbolic Vari-
      ables in Executable Sequence Charts, Proc. 17th Ann. AM Conf. on Object-
      Oriented Programming, Systems, Languages, and Applications (OOPSLA
      2002) 83-100
[18] Harel, D., Kugler, H., Marelly, R. and Pnueli, A., Smart Play-Out of Behavioral
      Requirements, Proc. 4th Int. Conf. on Formal Methods in Computer-Aided De-
      sign (FMCAD 2002) 378-398
[19] http://www.wisdom.weizmann.ac.il/mathusers/kam/CelegansModel/Demos.htm
[20] Sternberg, P. W.: Lateral inhibition during vulval induction in Caenorhabditis
      elegans Nature 335 (1988) 551-554
[21] Harel, D., A Grand Challenge for Computing: Full Reactive Modeling of a
      Multi-Cellular Animal, position paper, UK Workshop on Grand Challenges in
      Computing Research, Oct. 2002; available at
      http://www.wisdom.weizmann.ac.il /~dharel/papers/GrandChallenge.doc