Cell-DEVS/GDEVS for Complex Continuous Systems

Gabriel A. Wainer
Department of Systems and Computer Engineering
Carleton University
4456 Mackenzie Building
1125 Colonel By Drive
Ottawa, Ontario, K1S 5B6, Canada
Gabriel.Wainer@sce.carleton.ca

Norbert Giambiasi
Université d'Aix-Marseille III
Av. Escadrille Normandie Niemen
13397 Marseilles
Cédex 20, France

The Cell–Discrete Event System Specification (Cell-DEVS) formalism allows defining asynchronous cell spaces with explicit timing delays (based on the specifications of the DEVS formalism). The authors used Cell-DEVS to solve different applications and go one step further in the definition of complex continuous systems by combining Cell-DEVS and Generalized DEVS (GDEVS). They focus on a model describing the electrical behavior of the heart tissue, as previous research in this field has thoroughly studied this problem using differential equations and cellular automata. The authors show that they can provide adequate levels of precision at a fraction of the computing cost of differential equations. Their thesis is that the use of the GDEVS formalism is perfectly suited to attack problems such as this one, improving complex systems analysis. The authors show that their approach permits making models easily extensible to provide different actions in different cells while not affecting performance.

Keywords: DEVS models, Cell-DEVS models, GDEVS, cellular automata, discrete event simulation, heart tissue modeling, Hodgkin-Huxley model

1. Introduction

Complex systems analysis has been the object of study of researchers since the early ages of scientific development. A number of mathematical techniques have helped researchers to better analyze the systems under study; one of the preferred tools is the partial differential equation (PDE) formalism [1]. Unfortunately, in most complex systems, solutions to these equations are very difficult or impossible to find. Due to this reason, a variety of numerical methods were created to find approximate solutions to these equations, being successful in studying many different phenomena. The appearance of digital computers allowed the enhancement of previously existing numerical methods while enabling the creation of new techniques. Simulation-based approaches succeeded in providing the means of analyzing specific problems (instead of the general solutions obtained by solving PDEs), helping to solve problems with a level of detail unknown in earlier stages of scientific development.

Many of the simulation-based and numerical methods were based on extensions to the PDE formalism, but in the past 20 years, a radically different method has gained popularity. This technique consists of representing physical systems as cell spaces. The most common method for cellular computing, called the cellular automata (CA) formalism [2, 3], has been widely used to describe complex systems. A cellular automaton is organized as an n-dimensional infinite lattice of elements, each holding a state value and a very simple computing apparatus. The composite action of thousands of these cells can reproduce the behavior of complex physical systems. The composite activities of a CA define a global transition function that updates the state of the cell space through updates of discrete values in each cell. Cell states are changed by a local computing function, which uses the present value for the cell and a finite

DOI: 10.1177/0037549705052233

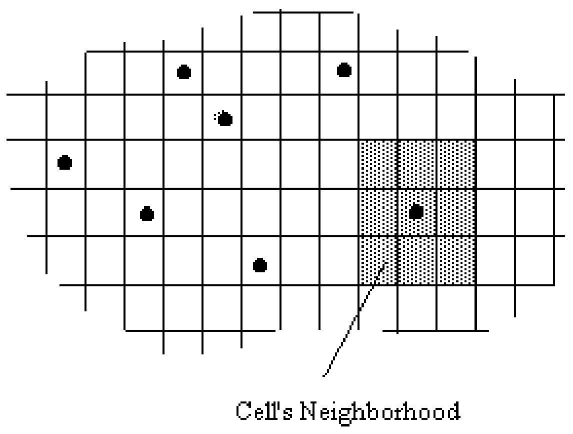


Figure 1. Sketch of a cellular automaton

set of neighbor cells, as shown in Figure 1. Conceptually, these local functions are computed synchronously and in parallel, using the state values of the present cell and its neighbors.

CA popularity has grown in the past few years (see [4, 5]), and recently it has received a tremendous impulse thanks to the works by Wolfram [6], which received special attention by the scientific community and the media. Despite these efforts, we have shown that CA has several problems that constrain power, usability, and feasibility to analyze complex systems [7]. The first problem we can identify in CA is related to the use of a discrete time base for cell updates. This constrains the precision and efficiency of the simulated models: to achieve higher timing accuracy, smaller time slots must be used, producing more demanding needs in terms of processing time. A second problem is that cellular automata, which are asynchronous in nature, must be implemented in digital computers, which often results in synchronous implementation. Furthermore, the discrete time implementation of the formalism makes it very difficult to handle time-triggered activity in each of the cells, which is usually required when defining complex applications.

We defined the Cell–Discrete Event System Specification (Cell-DEVS) formalism [8] to overcome these problems. Cell-DEVS allows defining asynchronous cell spaces with explicit constructions for the timing definition. This approach permits describing cell spaces as discrete event models, based on the formal specifications of the DEVS formalism [9]. In Cell-DEVS, each cell in a cellular model is seen as a DEVS atomic model, and a procedure for coupling cells is defined based on the neighborhood relationship. Explicit timing delay constructions can be used to define precise timing in each cell, which is defined by a local computing function combined with a delay construction (a sketch of Cell-DEVS models is presented in Fig. 2).

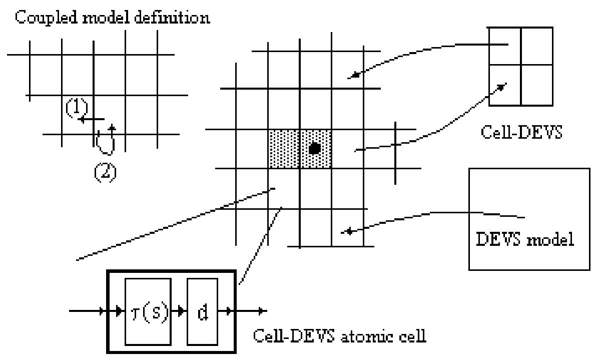


Figure 2. Informal definition of Cell-DEVS

These models are based on the specifications of the DEVS formalism. DEVS was defined in the early 1970s as a way of specifying discrete event systems organized hierarchically and using a modular description. A DEVS model is seen as composed by behavioral (atomic) submodels that can be combined into structural (coupled) models. As the formalism is closed under coupling, coupled models can be seen as new base models that can be integrated hierarchically. This strategy allows the reuse of tested models, allowing one to reduce development times.

DEVS models run asynchronously; consequently, every cell in a Cell-DEVS model runs asynchronously from others. Only the active cells in the cell space are triggered independently from any activation period. The hierarchical nature of DEVS also permits the integration of these cellular models with others defined using different formalisms, resulting in enhanced facilities for the modeling of complex systems.

Cell-DEVS enabled us to successfully solve a variety of complex problems in different areas [10-12]: biology (watersheds, fire spread, ant colonies), physics (crystal growth, lattice gases, heat diffusion), chemistry (flow injection analysis), and several artificial systems (autonomous robots, urban traffic, etc.). Nevertheless, even with the current advances in cell-based modeling and simulation techniques, problem solving using PDEs is still very popular. This responds to several facts:

- Most educational institutions around the world provide extensive instruction in this field.
- Thousands of existing models already have been developed using this approach, which results in an costly asset that most organizations using models for research or engineering are not willing to replace.
- Different tools and libraries provide facilities to solve PDEs.

In most cases, a great deal of resources was spent in this approach in terms of training, software development, and human resources. Hence, changing this approach to use representations that are more advanced is an unrealistic assumption for the near future.

Considering this scenario, we have focused on providing enhanced mechanisms for model definition based on cellular models integrated with PDEs. We are concentrating on physical systems that can be described as cellular models, and we want to improve the precision and execution speed of these models while using the current expertise of modeling specialists in different domains. We want to enable modelers to describe individual components of cellular models using PDEs approximations, which could result in enhanced model definition and would help to bridge the gap between traditional modeling techniques and cellular computing. We want to provide a means to defining cellular models in which individual cells use PDEs. Cell-DEVS models will be used to create cell specifications with timing delays, and each cell will run a smaller portion of a complex system of PDEs, embedded as rules in one cell. The researchers will be able to focus on defining smaller portions of a problem and expressing it using simpler differential equations, which can be solved easier than the complete system, creating a very precise model of each cell. The cell's timing delays can be used to define asynchronous behavior for each cell, and the resulting cellular models will be able to run asynchronously and in parallel, thus improving precision and performance.

We have put into consideration two important issues: how to keep the ability of CA to describe very complex systems using very simple rules (which is its main advantage) and how to bridge the gap between a continuous variable formalism such as PDEs and a discrete event description such as DEVS. Our thesis is that the use of the Generalized Discrete Event System Specification (GDEVS) formalism [13] attacks both of these problems simultaneously. GDEVS is a formalism for the specification of discrete event models of dynamic systems. The originality of GDEVS stems from the use of polynomials of arbitrary degree (as opposed to constant values) to represent the piecewise input/output trajectories of a discrete event model. In essence, GDEVS constitutes a generalization of the classical discrete event modeling approaches, including DEVS, in that a classical model may be viewed as a GDEVS model of order 0 (the trajectories are represented by a polynomial of order 0). Classical discrete event abstractions of dynamic systems are based on the mapping of piecewise constant input/output segments (obtained perhaps through threshold sensors) onto discrete events. GDEVS adopted a radically new approach based on a new definition of the concept of the event [14, 15]. In GDEVS, the target real-world system is modeled through piecewise polynomial segments. If we note that the polynomial coefficients have piecewise constant trajectories, we can build a discrete event abstraction in the coefficient space using the concept of a coefficient event. A coefficient event is thus considered as an instantaneous change of at least one value of the coefficients defining the piecewise polynomial trajectory of the considered variable. An event is a list of coefficient values defining the polynomial that describes the trajectory of the variable.

GDEVS will enable us to keep the complexity of the rules defining each cell's behavior to a minimum expression while still enabling the users to define their problems using PDEs. Using GDEVS to define the behavior for each cell will also enable us to provide adequate precision while incurring fewer time steps when compared with traditional numerical methods. It will also improve the accuracy obtained if we compare the results obtained by traditional CA due to the improved definition of model states. This approach also provides the advantage of traditional CA (e.g., the possibility of defining models that are very simple in terms of representation). Cell-DEVS enables the definition of specialized behavior in certain areas of the space, thus permitting modeling-modified phenomena in particular regions of the cell space. Such combined analysis is unfeasible using PDEs or CA. Likewise, the use of DEVS as the basic formal specification mechanism enables us to define interactions with models defined in other formalisms: individual cells can provide data to those models, and integration between them could enable defining complex hybrid systems.

We have successfully tested our approach for different complex systems, and here we introduce the use of the technique for modeling the electrical behavior in the heart tissue. Previous research in this field has studied this problem using PDEs and CA, and we show that we can provide adequate levels of precision at a fraction of the computing cost of the numerical methods employed to solve the PDEs. We also show that we can provide much more precise results than the ones previously obtained by CA. Finally, we show that models are easily extensible. The use of Cell-DEVS/GDEVS highly improved modeling activities, as the formal specification of the cell spaces helped reduce development and testing costs [7].

2. Background

A real system modeled using DEVS [9] can be described as being composed of several submodels, each being behavioral (atomic) or structural (coupled). Each of these basic models consists of a time base, inputs, states, outputs, and functions to compute the next states. New models can be integrated into a model hierarchy, allowing reuse of tested models, reducing testing time, and improving productivity. DEVS, as a discrete event formalism, uses a continuous time base, which allows accurate timing representation. DEVS also provides the advantages of being a formal approach: formal conceptual models can be validated, improving the error detection process and reducing testing time.

A DEVS atomic model can be formally described as follows:

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D > .$$

X is the input events set;

S is the state set;
Y is the output events set;
$\delta_{int} : S \to S$ is the internal transition function;
$\delta_{ext} : Q \times X \to S$ is the external transition function, where $Q = \{(s, e)/s \in S$, and $e \in [0, D(s)]\}$;
λ: $S \to Y$ is the output function; and
$D : S \to R_0^+ \cup \infty$ is the duration function.

Models use input/output ports to communicate. Each state in a model has a given lifetime, defined by the duration function. Once the lifetime of a given state is consumed, the internal transition function is activated to produce an internal state change. Before that, the output function is activated to generate the model's outputs. At any moment, a model can receive input external events from other models through its input ports. When an external event arrives, the external transition function is activated.

An atomic model can be integrated with other DEVS models to build a structural model. These models are integrated by other atomic or coupled models. They are formally defined as

$$CM =< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, select > .$$

X is the set of input events;
Y is the set of output events;
$D \in N$, $D < \infty$ is an index for the components of the coupled model, and $\forall \in D$;
M_i is a basic DEVS model;
I_i is the set of influencees of model i, and $\forall j \in I_i$;
$Z_{ij} : Y_i \to X_j$ is the i to j translation function; and
Select is the tie-breaking selector.

Each coupled model consists of a set of basic models (atomic or coupled) connected through the input/output ports of the interfaces. Each component is identified by an index number. The influencees of each model define other models where output values must be sent. The translation function uses an index of influencees, created for each model (I_i). The function defines which outputs of model M_i are connected to inputs in model M_j. When two submodels have simultaneous events, the *select* function defines which of them should be activated first.

The Cell-DEVS formalism extended this basic behavior to allow the implementation of cellular models. The cells are defined as atomic models, and they can be specified as follows:

$$TDC =< X, Y, S, \theta, I, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D > .$$

X is the set of external input events;
Y is the set of external output events;
S is the set of sequential states for the cell;
θ is the definition of the cell's state;
$I \in S^{\eta+\mu}$ is the set of states for the input events;
$d \in R_0^+, d < \infty$ is the transport delay for the cell;
$\delta_{int} : \theta \to \theta$ is the internal transition function;
$\delta_{ext} : Q \times X \to \theta$ is the external transition function, where Q is defined as $Q = \{(s, e)/s \in \theta \times I \times d$; $e \in [0, D(s)]\}$;
τ: $I \to S$ is the local computation function;
λ: $S \to Y$ is the output function; and
$D : \theta \times I \times d \to R_0^+ \cup \infty$ is the state's duration function.

A cell uses the input values I to compute its next state, which is obtained by applying the local computation function τ. A delay function associated with each cell enables one to defer the moment to transmit the computed result. There are two types of delays: inertial and transport. For the transport delay, the next value will be added to a queue sorted by output time, and the results will be stored during the delay. When this time is consumed, the value will be sent out. Inertial delays use a preemptive policy; that is, if the cell state changes before the delay, the previously computed result is not transmitted. This basic behavior is provided by the δ_{int}, δ_{ext}, λ, and D functions.

After the basic activity of a cell is defined, a whole cell space is built by creating a coupled Cell-DEVS model that includes copies of each of the atomic cells. The Cell-DEVS coupled model can be defined as follows:

$$GCC =< X_{list}, Y_{list}, X, Y, \eta, \{m, n\}, N, C, B, Z > .$$

$X_{list} = \{(k, l)/k \in [0, m], l \in [0, n]\}$ is the list of input coupling;
$Y_{list} = \{(k, l)/k \in [0, m], l \in [0, n]\}$ is the list of output coupling;
X is the set of external input events;
Y is the set of external output events;
$\eta \in N$ is the neighborhood size, and N is the neighborhood set;
$\{m, n\} \in N$ is the size of the cell space;
C defines the cell space, where $C = \{C_{ij}/i \in [1, m]$, $j \in [1, n]\}$, with $C_{ij} =< X_{ij}, Y_{ij}, S_{ij}, N_{ij}, d_{ij}, \delta_{intij}, \delta_{extij}, \tau_{ij}, \lambda_{ij}, D_{ij} >$ a Cell-DEVS atomic model;
B is the set of border cells, where

1. $B = \{\emptyset\}$ if the cell space is wrapped, or
2. $B = \{C_{ij}/\forall(i = 1 \vee i = m \vee j = 1 \vee j = n) \wedge C_{ij} \in C\}$, where $C_{ij} = < X_{ij}, Y_{ij}, S_{ij}, I_{ij}, d_{ij}, \delta_{intij}, \delta_{extij}, \tau_{ij}, \lambda_{ij}, D_{ij} >$ is a Cell-DEVS atomic model, if the border cells have different behavior than the rest of the cell space.

Z is the translation function, defined by

- $Z : P_{kl}^{Y_q} \to P_{ij}^{X_q}$, where $P_{kl}^{Y_q} \in I_{kl}, P_{ij}^{X_q} \in I_{ij}, q \in [0, \eta]$ and $\forall(f, g) \in N, k = (i + f) mod m; l = (j + g) mod n$;
- $P_{ij}^{Y_q} \to P_{kl}^{X_q}$, where $P_{ij}^{Y_q} \in I_{ij}, P_{kl}^{X_q} \in I_{kl}, q \in [0, \eta]$ and $\forall(f, g) \in N, k = (i - f) mod m; l = (j - g) mod n$;

select is the tie-breaking selector function, with $select \subseteq mxn \to mxn$.

Here, X_{list} and Y_{list} are input/output coupling lists used to define the model interface I. X and Y represent the input/output event sets. The space size is defined by $\{m, n\}$, and N defines the neighborhood shape. C, together with B, the set of border cells, and Z, the translation function, defines the cell space. The B set defines the cell's space border. If this set is empty, the space is "wrapped," meaning that cells in one border are connected with those in the opposite. In this case, every cell in the space will be considered as having identical activity. Otherwise, the border cells need to be provided with a behavior different from those of the rest of the model. Finally, the Z function allows one to define the coupling of cells in the model. This function translates the outputs of the mth output port in cell C_{ij} into values for the mth input port of cell C_{kl}. Each output port will correspond to one neighbor, and each input port will be associated with one cell in the inverse neighborhood. The ports' names are generated using the following notation: $P_{ij}^{X_q}$ refers to the qth input port of cell C_{ij}, and $P_{ij}^{Y_q}$ refers to the qth output port. These ports correspond with the port names denoted as X_q or Y_q for each cell.

We developed our studies using the CD++ tool kit [16], which was built following the formal specifications of DEVS and Cell-DEVS, as described in this section. The tool provides a specification language to describe the behavior of each cell and the global parameters for the coupled cell space (including size, influencees, neighborhood, and borders). Using these parameters, a complete Cell-DEVS is built using the formal specifications described earlier. The activities of a cell are defined using rules with the following form:

VALUE DELAY { CONDITION }

Each rule indicates that, if the CONDITION is satisfied, the state of the cell will change to the designated VALUE, and this new state value will be spread to the neighboring cell after the chosen DELAY. If the condition is not valid, the next rule is evaluated (according to the order in which they were defined), repeating this process until a rule is satisfied. A neighborhood, which is defined as a list of offsets from the current set, can be composed of nonadjacent cells, and the neighborhood's dimension can be similar or inferior to the model's dimension. Space zones, defined by a cell range, can be associated with a set of rules different from the rest of the cell space. Common operators are included: Boolean (AND, OR, NOT, XOR, IMP, and EQV), comparison (=, !=, <, >, <=, and >=), and arithmetic (+, –, *, and /). In addition, different types of functions are available: trigonometric, roots, power, rounding and truncation, module, logarithm, absolute value, minimum, maximum, greatest common denominator (GCD), and least common multiple (LCM). Other existing functions allow one to check if a number is integer, even, odd, or prime. Some functions allow one to query the cell state of the neighborhood: *truecount*, *falsecount*, *undefcount*, and *statecount*(n). Common constants are defined as follows: pi, e, and certain constants used in the domains of physics and chemistry (gravitation, acceleration, light speed, Planck's, etc.). The time function returns the global simulated time. Other functions allow one to obtain values depending on the evaluation of a certain condition. $IFU(c, t, f, u)$ evaluates the c condition, and if it is true, it returns the t value. It returns f if it is false and u if it is undefined. On the other hand, the function $IF(c, t, f)$ returns t if c evaluates to true and f otherwise. Finally, several functions are used to generate pseudo-random numbers using different probability distributions.

We intend to extend the basic behavior provided by Cell-DEVS atomic models to permit users to specify cells with continuous variable behavior using the GDEVS formalism. GDEVS considers the general case of dynamic systems with piecewise continuous input/output trajectories, and it has solved how to transform these piecewise continuous trajectories into discrete event trajectories. This transformation was done by achieving a partition of the output trajectory into piecewise polynomial segments. To each of these output segments corresponds a continuous segment of the state trajectory and piecewise constant segments in the space of polynomial coefficients. In a GDEVS model, an event is an instantaneous change in at least one of the values of the coefficients of the polynomial describing the signal.

For example, the continuous signal presented in Figure 3a can be approximated by the piecewise linear segment of Figure 3b or by the events of order 1, shown in Figure 3c (discrete event abstraction). If we identify a trajectory $w < t0; tn > \rightarrow A$ as a trajectory on a continuous time base, characterized as a finite set of instants $\{t_0, t_1, \ldots, t_n\}$ associated with constant pairs $(a_i; b_i)$ such that $\forall t \in \ < t_i; t_j >, w(t) = a_i t + b_i$, and $w < t_0; t_n >= w < t_0; t_1 > *w < t_1; t_2 > * \ldots * w < t_{n-1}; t_n >$ (where * represents the operator left concatenation of segments), then we can build a piecewise trajectory such as the one introduced in Figure 3b. By using higher or lower order polynomial approximations, we obtain GDEVS models with different coefficient events.

In our proposal, the behavior of each cell in a Cell-DEVS model is described using GDEVS. In previous experiences, we were able to include continuous functions in each cell. Nevertheless, the definition was constrained to defining discrete time versions of the PDEs running in each of the cells. This prevented two of the main advantages of using Cell-DEVS: the use of discrete events and the specification cellular models as a composite of cells described with very simple rules. If we apply a cell-based approach, we might be able to express the continuous functions using ad hoc simple rules. Nonetheless, most researchers would still prefer to use a PDE in each cell, which would result in performance degradation. We will show how Cell-DEVS models whose components are defined using GDEVS (see Fig. 4) can overcome these problems. The idea is to approximate a PDE in the local computing function τ by using a GDEVS of the desired precision. This will provide a means for improved performance while having simpler rules for model definition (namely, concatenation of polynomials).

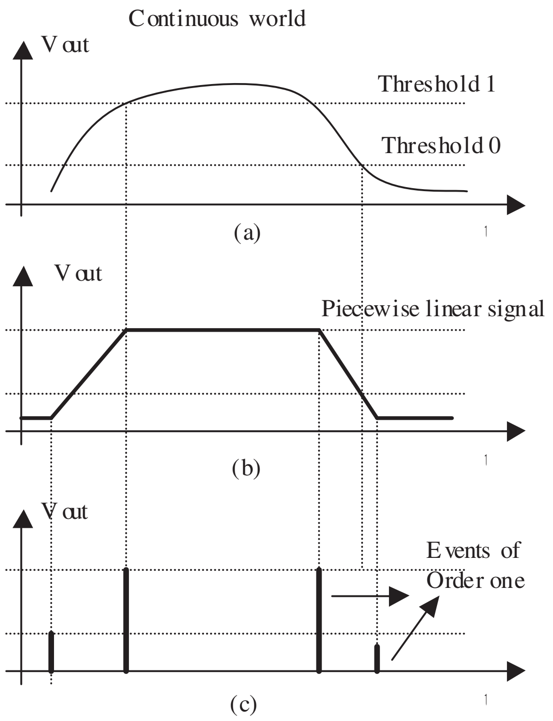


Figure 3. GDEVS approximation of a continuous signal: (a) continuous segment, (b) piecewise linear segment, and (c) first-order model

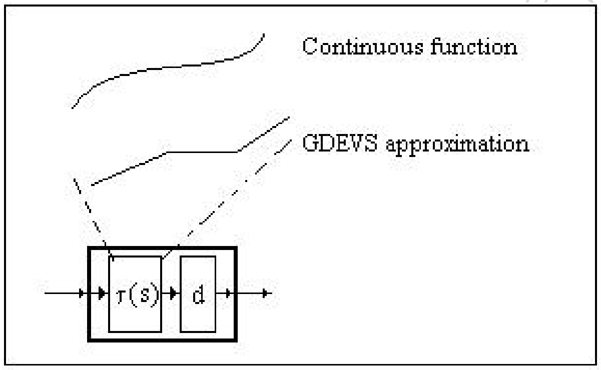


Figure 4. GDEVS approximation of Cell-DEVS local computing functions

The ideal case in terms of performance is when a linear approximation is able to provide high precision and bounded error, as linear models have low cost of execution and easy definition. Higher precision can be achieved by using higher order polynomials, with the cost of execution time and increased complexity of the rules defined. We also show that defining GDEVS models using polynomials of order 0 results in automatic definition of traditional CA.

In the remaining sections, we show how to apply these ideas to a well-known model describing the electrical behavior of the heart tissue, which has been solved using different techniques. This permits us to show the power of our approach while comparing the results with other existing formalisms.

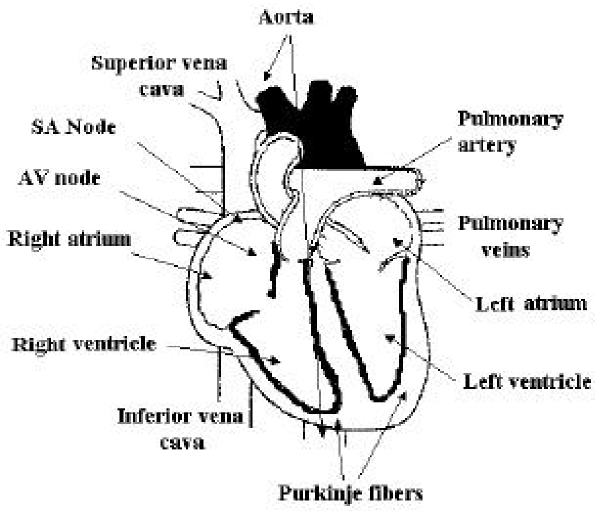


Figure 5. Basic anatomy of the heart

3. Modeling Behavior of the Heart Tissue

The heart is a muscle responsible for pumping blood into the circulatory system. The behavior of the phenomena occurring in the heart muscle and tissue has been extensively studied, and it has been reported in a wide variety of medical treaties (see, e.g., [17, 18]). In these documents, heart activity is usually analyzed according to three kinds of activities: mechanical, electrical, and cellular.

In terms of *mechanical* activities, the blood returns to the heart through the vena cava superior and inferior and flows to the right atria. The blood flows to the right ventricle, where it is pumped to the lungs to return oxygenated to the left atria. Then, it flows to the left ventricle, which returns the oxygenated blood to the body through the aorta. This is presented in Figure 5.

Mechanical activity is triggered by the *electrical* activity of the cells. The heart muscle is excitable, and the cells in its tissue respond to external stimuli by contracting the muscular cells. If the stimulus is too weak, the muscle does not respond; instead, if the voltage received is adequate, the cells contract at maximum capacity. The electrical conduction system of the heart is responsible for the control of its regular pumping. This activity originates in the sinoatrial (SA) node, also known as the *pacemaker*. This is an electrically active region of the heart that self-activates. Cells in the heart tissue are excited when adjacent cells are charged positively. In that case, an upstroke of its action potential is provoked, which will spread to nearby cells.

All excitable tissue, once activated, exhibits a refractory period before returning to rest. During the contraction period, the muscle is refractory and does not respond to external stimuli. Before starting a new contraction, the previous one should have finished, and shortly after the contraction, the muscle is relatively refractory. In this case, minimum stimuli do not generate response, but a stronger stimulus is able to generate a response. The electrical activity is started in the SA node, and it spreads through the atria muscle at a speed of 1 m/sec (for human beings, 80 msec are needed to activate the atria). After, the electrical activity is spread to the atrioventricular (AV) node, where it propagates slowly (0.1 m/sec), and then the excitation travels at 2 m/sec through the Purkinje fiber.

This electrical activity is originated by the *cellular* activities, which consist of the interchange of potassium and sodium ions in the walls of the cells. This chemical reaction produces potential differences of mV, which trigger the electrical activity. This behavior of cell membrane activity was originally characterized by Hodgkin and Huxley [19], in a foundational article that presented the detailed behavior of the intermembrane action potential function. They recognized different phases in this function:

1. The heart tissue is relaxed, and the interior of the membrane is electrically negative in relation to the surface, with a difference of potential of 50 mV.

2. The surface membrane is repolarized, creating two zones with a potential difference.

3. Electrical activity starts, and the external surface becomes negative, with a potential difference of 30 mV. This phase is called excitation (or depolarization).

4. Finally, negative voltage on the surface trespasses the membrane, and the original status is recovered. This phase is called repolarization.

The Hodgkin-Huxley model showed that virtually all membrane current models can be defined by writing the total membrane current, which is a sum of the individual currents carried by different ions through specific channels in the cell's membrane. The calculation is based on sodium ion flow, potassium ion flow, and the leakage ion flow. This behavior can be defined as follows:

$$I = m^3 h G_{Na}(E - E_{Na}) + n^4 G_K(E - E_K) + G_L(E - E_L). \quad (1)$$

I is the total ionic current across the membrane,
m is the probability that one particle contributed to activate the sodium gate,

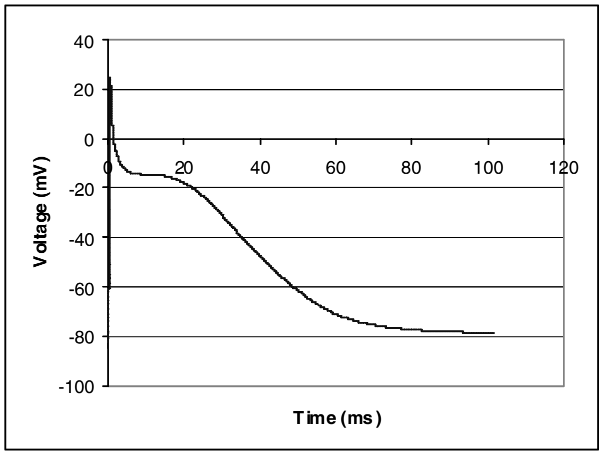


Figure 6. Action potential in the atria cells using Hodgkin-Huxley equations

h is the probability that one inactivation particle has not caused the sodium gate to close,
G_{Na} is the maximum sodium conductance,
E is the total membrane potential,
E_{Na} is the sodium membrane potential,
n is the probability that one of four particles influenced the potassium gate,
G_K is the maximum possible potassium conductance,
E_K is the potassium membrane potential,
G_L is the maximum leakage conductance, and
E_L is the leakage membrane potential.

Hodgkin and Huxley [17] computed empirical formulas for the sodium gate activation (m), sodium particle activation probability (h), and potassium gate activation probability (n). They also found the values of the remaining parameters of equation (1), which where shown to be constant. By applying the Hodgkin-Huxley equations, we can obtain the action potential function for the cells in different regions of the heart tissue. The behavior of different cells can be defined by variation in conductivity, length of the fibers, and so on. The authors also showed that the results of this equation are equivalent to the results found in experimental data. For instance, Figure 6 shows the results obtained when using the Hodgkin-Huxley equations, using parameters corresponding to cells of the atria. We use this example in the following sections to build a Cell-DEVS/GDEVS model of the heart tissue and to compare the results obtained with other approaches.

4. Modeling Heart Tissue as Cell-DEVS Models

The Hodgkin-Huxley model has been extensively used in different studies, as it has been shown that it reproduces with fidelity the electrical properties in the myocardium cells. Nevertheless, whereas solving this equation using

```
[Heart]
type : cell
dim : (5,5)
delay : transport
border : nowrapped
neighbors : (-1,-1) (-1,0) (-1,1) (0,-1)
neighbors : (0,1) (1,-1) (1,0) (1,1) (0,0)
localtransition : Heart-rules

[Heart-rules]
rule : 2            0.48   {(0,0)=0 and statecount(2)>0 }
rule : 1            1.48   { (0,0) = 2 }
rule : 0            17.5   { (0,0) = 1 }
rule : { (0,0) }    0      { t }
```

Figure 7. Cell-DEVS definition of a simple heart tissue model

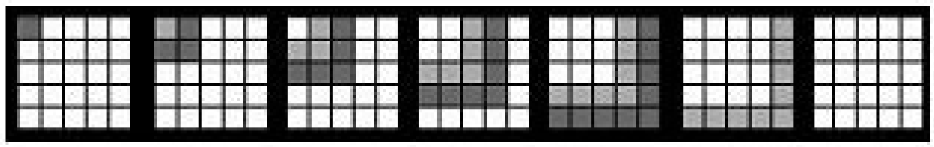


Figure 8. Heart tissue model execution

numerical methods for one cell is feasible, the use of this model in a realistic reproduction of the heart tissue (probably consisting of millions of cells) can be computationally expensive. Consequently, different authors have tried to simplify the complexity of the equations, and various studies have tried to solve this problem using CA (see, e.g., [20-22]). Most of these models are based on simple CA for excitable media, which discretize the Hodgkin- Huxley results.

In Ameghino and Wainer [10], we used Cell-DEVS to build a discrete variable model of heart tissue conduction. In this model (which uses a similar approach to other models built using CA), we recognized three states for a cell: resting, excited, or recovering. We defined this model in the CD++ tool kit, and Figure 7 includes a complete specification of it.

Model definition begins by defining the Cell-DEVS coupled model and its parameters: size (5×5 cells), neighborhood shape (all of the adjacent cells), kind of delay (transport, as we want every state change to be transmitted without preemption), and borders (this is a nonwrapped model, and special rules were defined for the borders). The heart rules section represents the local computing function for the model. Here, the first rule represents the initiation of electrical activity in a resting cell (with value 0). In that case, we check to see if any of the neighbors is excited (value 2). In that case, the cell is excited. Second and third rules define the cells changing to the recovering and resting states. The last rule states that in every other case (*t* means "true"), the cell keeps its present state. Figure 8 shows the results obtained when this model executes. It shows the evolution of this with a pacemaker cell in (0,0).

As we can see, the model represents the tissue action using very simple rules, which has several advantages. Cell behavior is defined using simple rules, which makes it easy to modify the existing model to experiment with different conditions. We also see that delay functions are associated with each of the rules representing each state cell. When describing this model using CA, timing definition is more complex, and it can result in extensive simulation time to achieve the desired precision. Likewise, any changes in the delay functions can result in complex changes in the CA definitions. Instead, Cell-DEVS timing delays can provide complex timing description using rules that are straightforward to define. For instance, this model represents three different delays at different scales. To achieve such precision in CA, we should choose the smallest timeslot for simulating time advance. Instead, in Cell-DEVS, each rule is triggered by an event that is executed asynchronously in each of the cells at randomly chosen instants.

Although representing this problem as CA permits introducing simple rules, it poses a problem in the model's precision. As we can see, we have discretized the continuous function shown in Figure 6 with only three different discrete states. This could seriously affect the execution results of the model if we needed to introduce modifications to the cell's standard behavior. For instance, arrhythmias affect isolated groups of cells, modifying the shape of the action potential curve, which could require defining a completely erratic behavior for a group of cells. Another

```
[heart]
type : cell
dim : (5,5,2)
delay : transport
border : nowrapped
neighbors : (-1,-1,0) (-1,0,0) (-1,1,0)
neighbors : (0,-1,0)  (0,0,0)  (0,1,0)
neighbors : (1,-1,0)  (1,0,0)  (1,1,0)  (0,0,1)
localtransition : heart-rule-AP

[heart-rule-AP]
rule : { AP(cellpos(0) } 1 { cellpos(2)=0 and (
                           (-1,0,0) > 0 or (0,-1,0) > 0 or (-1,-1,0)>0) and (0,0,0) = -
83.0) }
rule : { AP(cellpos(0) } 1 { cellpos(2)=0 }
rule : { if( (0,0,0) = 1.0 or (0,0,0) = -83.0, 0.0, 1.0) } 1 { cellpos(2)=1 }
```

Figure 9. Cell-DEVS definition of the action potential function for a heart tissue model [23]

example considers analysis of the cell's behavior during the refractory period: if enough voltage is received on a cell, the cell is excited, but if the voltage is not enough, it will ignore the stimuli. Representing this behavior with CA is very difficult; instead, if a PDE is included on each cell, it will be able to adequately react to each possible modification of the parameters.

As a result, we decided to implement this model as Cell-DEVS, running the Hodgkin-Huxley model in each of the cells. We implemented a model of the action potential function for the cells in the heart atria [23]. This Cell-DEVS model simulates the electrical behavior of the cells, following the Hodgkin-Huxley model, as described in section 3, discretizing time in each of the cells under execution. Figure 9 shows the model definition using CD++.

We first define the size of the cell space, which, in this case, is a 3D model with $5 \times 5 \times 2$ cells. This model uses a transport delay, and it is nonwrapped (we define specialized behavior for the cells in the border). The following lines in the specification define the neighborhood shape (in this case, all the adjacent cells in plane 0 and the upper cell, which will be used to define whether the current cell should be computed). Then, we define the local computing function, called *heart-rule-AP*. The function is defined by two rules. The first one will be evaluated only by the cells in the first plane in the model (*cellpos*(2) = 0) and only if the cell is resting and a positive voltage is detected in the cell's neighborhood. This rule will trigger the update of the cell state using the Hodgkin-Huxley equations presented in section 3. The second rule will be used in the subsequent activations. The third rule is evaluated only by the second plane (*cellpos*(2) = 1), and it is used to trigger time-based actions for the first plane. This plane is just changing its state from 0 to 1 and vice versa in each time step, triggering the execution of the rules of the action potential function in plane 0. This is needed because Cell-DEVS only considers activation of a cell under asynchronous events, and if no event is created, the cell goes to a quiescent state, which is avoided by this rule.

The AP function in this model receives the coordinates of the current cell and its current state. Using these values, it recovers the previous state of the current cell and computes the next voltage using equation (1). Figure 10 shows the execution results of this model. As we can see, the results obtained are the same as those we obtained earlier by solving analytically the Hodgkin-Huxley equations (in fact, most of the source code originally developed to build the AP function was reused in this Cell-DEVS model).

As we can see, this model improves precision over the CA, and thus we are able to define advanced cell behavior easily. For instance, by activating the AP function with different parameters in different cells, we are able to reproduce the activity in sick cells (e.g., those with arrhythmias, fibrillation, or conductivity problems). Nevertheless, this model is very expensive in terms of computing resources.

5. Using Cell-DEVS/GDEVS to Improve Model Definition

Having defined the heart tissue model using two traditional approaches (i.e., CA and Hodgkin-Huxley equations), we attacked the problem using Cell-DEVS/GDEVS. The first step in this study was to find a polynomial approximation to the original PDE defining the cell's behavior. Figure 11 shows the result of this approximation function.

We approximated the initial equation experimental data using eight polynomials of degree 1 to build a GDEVS model of order 1. A higher level of accuracy can be obtained using GDEVS of a higher level with the same number of states and events to treat. In the present case, the identification of the parameters in each of the polynomials was obtained by minimizing a quadratic criterion using minimum squares. The polynomials we used in Figure 11 are defined by

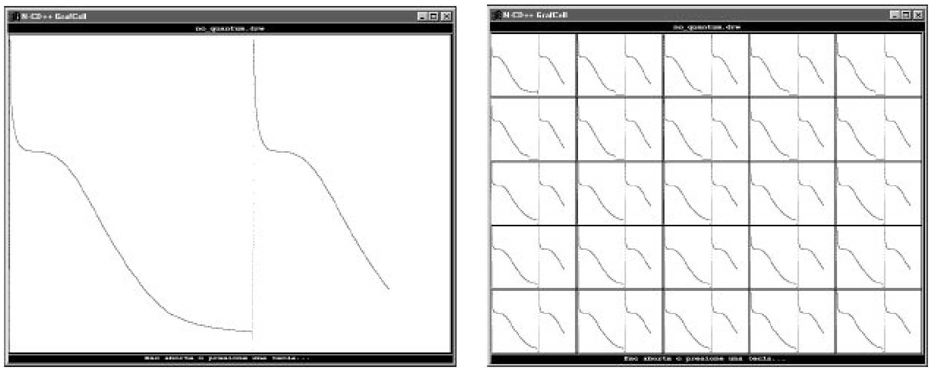


Figure 10. Model execution using Hodgkin-Huxley equations: (a) individual cell and (b) cell space

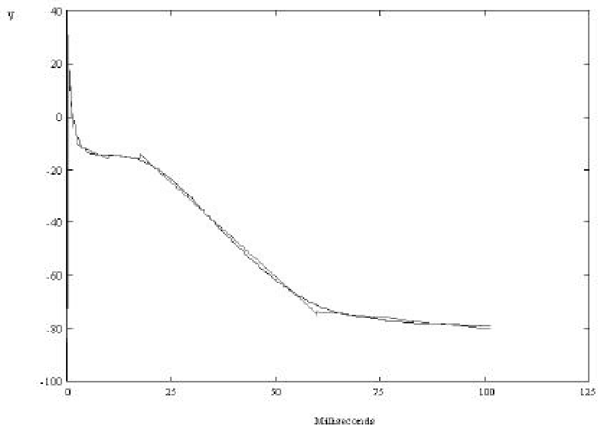


Figure 11. Linear approximation of the action potential function

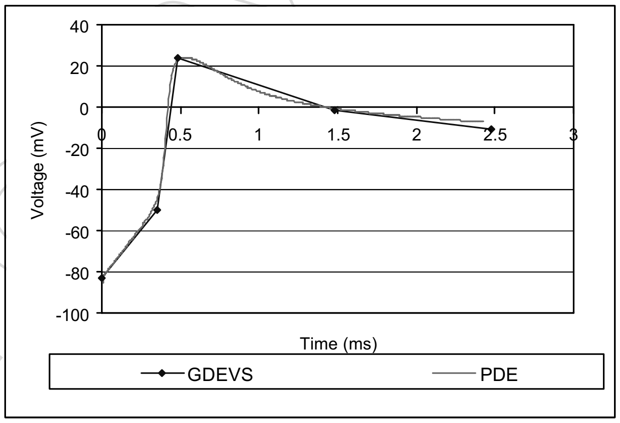


Figure 12. Approximation of the action potential function: action triggering

$$P_i(t) = a_i t + b_i \qquad \forall i \in [1, 8]$$

using the coefficients presented in Table 1.

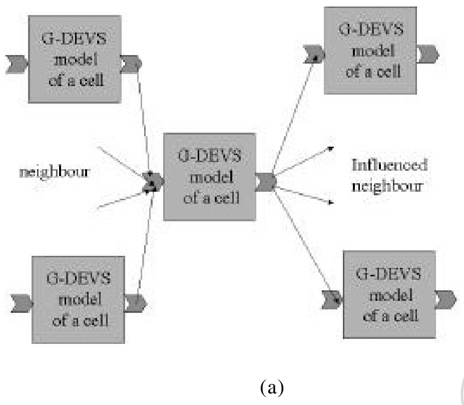
Although the original function appears to be simple, we needed to use eight polynomials. This was due to the fact that, when the cell is triggered, the signal generated by the Hodgkin-Huxley model is nonlinear, as we can see in Figure 12. Thus, between 0 and 2 msec were needed to approximate the action potential using four different polynomials (as shown in Table 1). We also introduced an intermediate state in which the polynomial evaluation would result in obtaining a positive value, which will trigger activity in the neighboring cells in this example (polynomial $P2$ is in charge of this).

When using GDEVS for this model, we need to transform the coefficients in the polynomials into discrete event signals, as explained in section 2. Each cell will use polynomial coefficients to compute the current state and to inform the cell's state to the neighbors, as shown in Figure 13. The specification of the local computing function included in each of the cells will now receive the current coefficient from the neighboring cells, as shown in Figure 13a. Each cell is now defined as shown in Figure 13b, and it will receive the coefficients of the neighbors by an event of order 1, which will be used to compute the state of the cell. The cell's outputs will now be the current cell states specified as polynomial coefficients. Timing of activation for each polynomial can be easily defined using the model delay functions.

Using these ideas and the polynomial definitions in Table 1, we can now define the actions of each of the cell's local computing functions, which are described by the state graph in Figure 14. The figure defines a GDEVS model with the classical state transition functions using an event

Table 1. Polynomial coefficients for the action potential model

i	*ai*	*bi*	Time (ms)
1	1.0250	–83.1478	[0, 0.35)
2	6.4555	–275.5886	[0.35, 0.43)
3	–0.2765	37.4703	[0.43, 0.48)
4	–0.0661	8.7840	[0.48, 1.48)
5	–0.0073	–8.6492	[1.48, 2.48)
6	–0.0022	–12.1344	[2.48, 9.98)
7	–0.0143	10.6898	[9.98, 17.48)
8	–0.0016	–64.0617	[17.48, 60)
9	–0.0016	–64.0617	[60, +∝)



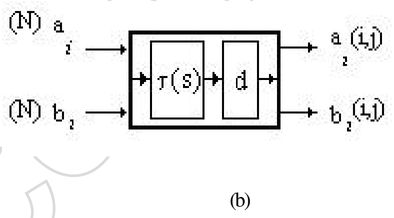


Figure 13. GDEVS cell specification: (a) model interconnection and (b) cell input data

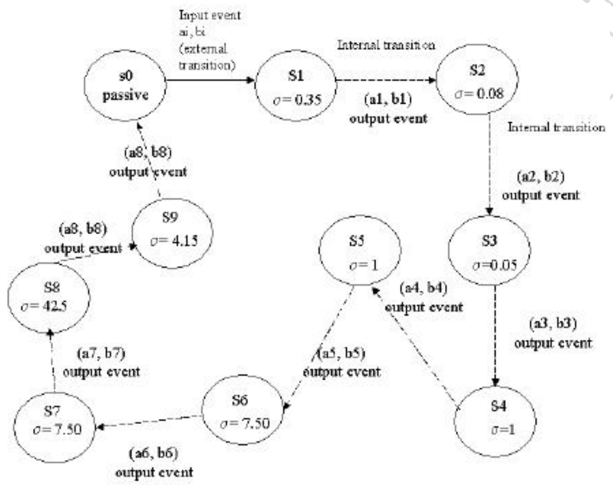


Figure 14. GDEVS specification of a cell

of order 1. The figure represents internal transitions with dotted lines and external transitions with solid lines.

As we can see, the cell is inactive until it receives an external stimulus from a neighboring cell. In that case, the cell is activated, and it produces internal state changes (represented by the coefficient in the polynomials, which are transmitted to the neighboring cells after the delay). The model flows through eight different states represented by each of the polynomials, plus an extra state to put the model into a resting state.

This specification will generate an output trajectory similar to the one described by the linear approximation in Figure 11. As we can see, this highly improves model precision at a low cost, in terms of both execution time and ease of modeling. We used the cell's specification in Figure 14 to define this model using the CD++ tool kit and compared the results obtained with those shown in section 4. Figure 15 shows the model implementation for a cell space in which each cell is created using the state machine in Figure 14.

This specification starts by defining the size of the cell space (6 × 6) and the remaining parameters needed by a Cell-DEVS specification—in this case, transport delays, a nonwrapped model, and the neighborhood shape, which includes all the adjacent cells. Then, we define the local computing function, *heart-rule-GDEVS*. This local computing function follows the specification in Figure 14. If a stimulus is received when the cell is inactive ((0,0) = −83), it will check the voltage received from the cells in the neighborhood (which is received through ports *ai* and *bi* and is computed by the *voltage* function) reacting to

```
[heart-GDEVS]
type : cell
dim : (6,6)
delay : transport
border : nowrapped
neighbors : (0,-1) (0,0) (-1,0) (-1,-1)
neighbors : (0,1)  (1,0) (-1,1) (1,1) (1,-1)
localtransition : heart-rule-GDEVS

[heart-rule-GDEVS]
rule : { S1 }                         0     {(0,0)=-83 and voltage(0,-1) > 0 or voltage(-1,-1) > 0 or
                                                                            voltage(-1,0)>0 }
rule : { S2, send(1.0250,-83.1478)    }  0.35   { (0,0) = S1 }
rule : { S3, send(6.4555,275.5886)    }  0.08   { (0,0) = S2 }
rule : { S4, send(-0.2765,37.47)      }  0.05   { (0,0) = S3 }
rule : { S5, send(-0.0661,8.784)      }  1      { (0,0) = S4 }
rule : { S6, send(-0.0073,-8.6492)    }  1      { (0,0) = S5 }
rule : { S7, send(-0.0022,-12.1344)} 7.50       { (0,0) = S6 }
rule : { S8, send(-0.0143,10.6898)    }  7.50   { (0,0) = S7 }
rule : { S9, send(-0.0016,-64.0617)} 42.5       { (0,0) = S8 }
rule : { S0, send(-0.0016,-64.0617)} 4.15       { (0,0) = S9 }
rule : { (0,0) }                      0     { t }

[voltage-function]
voltage(cellpos) = cell.ai * time + cell.bi
```

Figure 15. Cell-DEVS/GDEVS implementation of the heart tissue model

positive voltage in any of them. It will change to the corresponding state (*Si*, to the left of the specification) and will send the current *ai*, *bi* coefficients to the neighboring cells after the consumption of the delay. Each of the rules represents a cell's state change and the spread of the coefficients to the neighbors. Each of the cells will repeat the behavior defined here while storing the voltage value for display, which is shown in Figure 16.

As we can see, we obtained an output trajectory that is more precise than the one obtained with CA, which is defined by the output trajectory presented in Figure 11. This gain of precision involved only a low extra cost in terms of computing time. Likewise, the complexity added to the cellular model developed in Cell-DEVS/GDEVS is reduced when compared with the solution using PDEs (which required implementing the Hodgkin-Huxley functions of section 3).

The results of our experience are summarized in Figure 17. This figure shows the number of messages involved in simulating the heart tissue model using different approaches discussed here. We computed the number of messages issued in a Cell-DEVS/GDEVS model using a simple set of rules (*CDSimple*, such as the one shown in Figs. 7 and 8) and a second Cell-DEVS/GDEVS model with a larger number of intermediate states (*CD*, permitting better precision in the results obtained).

The figure also compares the results obtained with traditional CA (following the rules presented in Fig. 7) and the results of using numerical approximations for the Hodgkin-Huxley PDEs (with two different discretization steps to change the precision of the model approximation). We can see that the number of messages involved grows exponentially with the number of cells, but we can see that Cell-DEVS/GDEVS, which provides a much more precise signal, only reduces performance less than 5% when compared with traditional Cell-DEVS models. Cellular automata take longer, as every cell must be activated in every time step. Therefore, for a small number of cells, the execution time remains controlled, but when large cell spaces are considered, performance degrades.

Furthermore, GDEVS approximation highly improves not only performance when compared with traditional numerical methods but also precision when compared with traditional cellular computing. This can be seen in Figure 18a, in which we compare the average error in the heart tissue model for CA (or Cell-DEVS) models with a small number of states (*CD simple*), CA (or Cell-DEVS) with a larger number of states (*CD/CA*), and Cell-DEVS/GDEVS (*GDEVS*). We see that the cost of running Cell-DEVS/GDEVS models is minimum when compared with Cell-DEVS or CA models (the figure shows the average error for the 50×50 model of Fig. 17). Figure 18b shows the reason why the precision of Cell-DEVS/GDEVS models is higher. As we can see, each cell with GDEVS approximates adequately the original signal, whereas discrete variable models (such as CA) introduce a larger amount of error. These discrete values simplify the basic definition of the model but can make it difficult to detect state values with significance in terms of defining the other. In this figure, we can also see that the standard CA definition (like the one defined in Fig. 7) can be simply represented as a GDEVS model with a polynomial of order 0, which will create a piecewise continuous output trajectory.

Our approach has several other advantages:

- If higher precision is required, we can approximate the function by a higher degree polynomial.
- We can easily modify the model to represent other phenomena (e.g., abnormal activity in the cells can

```
Line : 83 - Time: 00:00:00:000
             0           1           2           3           4           5
   +-----------------------------------------------------------------------+
  0|   1.97000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  1| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  2| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  3| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  4| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  5| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
   +-----------------------------------------------------------------------+

Line : 115 - Time: 00:00:00:043
             0           1           2           3           4           5
   +-----------------------------------------------------------------------+
  0|   1.97000     1.99791   -83.00000   -83.00000   -83.00000   -83.00000|
  1|   1.99791     1.99791   -83.00000   -83.00000   -83.00000   -83.00000|
  2| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  3| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  4| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  5| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
   +-----------------------------------------------------------------------+

Line : 199 - Time: 00:00:00:086
             0           1           2           3           4           5
   +-----------------------------------------------------------------------+
  0|  24.19800    24.19800     1.99791   -83.00000   -83.00000   -83.00000|
  1|  24.19800    24.19800     1.99791   -83.00000   -83.00000   -83.00000|
  2|   1.99791     1.99791     1.99791   -83.00000   -83.00000   -83.00000|
  3| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  4| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  5| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
   +-----------------------------------------------------------------------+

...

Line : 420 - Time: 00:00:00:148
             0           1           2           3           4           5
   +-----------------------------------------------------------------------+
  0|  -0.99880    -0.99880     1.99791     1.99791   -83.00000   -83.00000|
  1|  -0.99880    -0.99880     1.99791     1.99791   -83.00000   -83.00000|
  2|   1.99791     1.99791     1.99791     1.99791   -83.00000   -83.00000|
  3|   1.99791     1.99791     1.99791     1.99791   -83.00000   -83.00000|
  4| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  5| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
   +-----------------------------------------------------------------------+

...

Line : 1087 - Time: 00:00:01:748
             0           1           2           3           4           5
   +-----------------------------------------------------------------------+
  0| -14.30660   -14.30660   -14.30660   -14.30660   -14.30660     1.99791|
  1| -14.30660   -14.30660   -14.30660   -14.30660   -14.30660     1.99791|
  2| -14.30660   -14.30660   -14.30660   -14.30660   -14.30660     1.99791|
  3| -14.30660   -14.30660   -14.30660   -14.30660   -14.30660     1.99791|
  4| -14.30660   -14.30660   -14.30660   -14.30660   -14.30660     1.99791|
  5|   1.99791     1.99791     1.99791     1.99791     1.99791     1.99791|
   +-----------------------------------------------------------------------+

...

Line : 1601 - Time: 00:00:10:153
             0           1           2           3           4           5
   +-----------------------------------------------------------------------+
  0| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  1| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  2| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  3| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  4| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
  5| -83.00000   -83.00000   -83.00000   -83.00000   -83.00000   -83.00000|
   +-----------------------------------------------------------------------+
```

Figure 16. Cell-DEVS/GDEVS execution of the heart tissue model

obtain by changing parameter values and defining a Cell-DEVS/GDEVS zone with the new behavior).

- We can define a more general model by making the slope and gradient (ai, bj) of each state to be defined as external parameters.
- We could easily modify the model specification to analyze more complex circumstances—for instance, inadequate excitation of a cell due to deformation to the action potential, detailed analysis of the activities of the Na/K channels, and so on. This new behavior can be modeled by simply modifying the rules described in Figure 15.

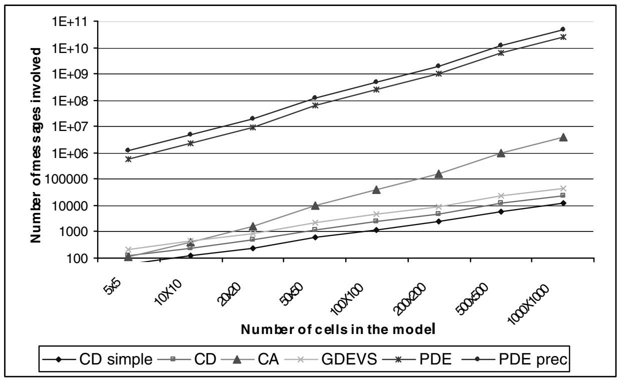
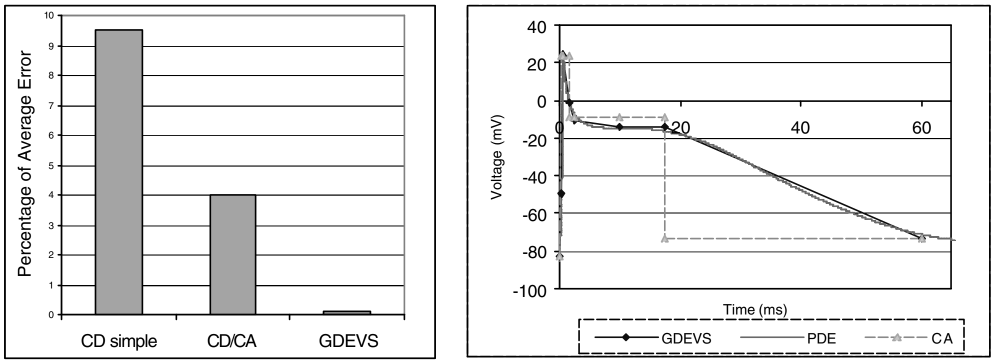


Figure 17. Comparing simulation models (logarithmic scale)

6. Conclusion

We showed how to combine Cell-DEVS and GDEVS to build very complex systems. The Cell-DEVS formalism allows (based on DEVS specifications) one to define asynchronous cell spaces with timing delays. Here, we combined Cell-DEVS with GDEVS models, permitting the definition of complex continuous systems easily. Cell-DEVS/GDEVS permits enhancing the modeling activities as the automatic definition of cell spaces is allowed, simplifying the construction of new models, and easing the automatic verification of the structural models. In this way, efficient development of complex models can be achieved.

We focused on the Hodgkin-Huxley model of the electrical behavior of heart tissue and compared the results obtained against those originally built with PDEs and cellular automata. We showed that we can provide adequate levels of precision at a fraction of the computing cost of differential equations. We proved that the GDEVS formalism is perfectly suited to attack problems such as this one, improving complex systems analysis. GDEVS uses polynomials of arbitrary degree, as opposed to constant values, to represent the piecewise input/output trajectories of a dynamic system. We also showed that our approach permits making models easily extensible to provide different actions in different cells. The hierarchical nature of the DEVS formalism permits attacking different levels of abstraction, which permits, for instance, building more detailed models about the behavior of ion interchange within each of the cells in the system. Likewise, the definition of different phenomena in groups of cells is straightforward, in terms of both Cell-DEVS and GDEVS specifications. Finally, the ability of Cell-DEVS to define delays with explicit constructions made the classification of complex timing easy.



(a) (b)

Figure 18. Analyzing model error: (a) average error and (b) a comparison of output trajectories

At present, we are working on the definition of other complex models using this approach (mainly, a fire spread model and a watershed formation system). This will provide us with a variety of different models, enabling us to start detailed studies on characterizing the error of this approach. We are also starting some work in related areas—namely, quantized DEVS models [24] and derived research—as we obtained good results in modeling continuous systems using quantized Cell-DEVS [25]. Nevertheless, at this moment, this effort is in a preliminary stage, as research is still required to simplify rule definition for quantized models. For instance, we are currently working on finding quantized versions of the Hodgkin-Huxley equations, which proved not to be as simple as finding GDEVS approximations. This particular area requires a great deal of effort to facilitate any future developments in building complex continuous systems using DEVS-based approaches.

7. References

[1] Taylor, M. 1996. *Partial differential equations: Basic theory*. New York: Springer Verlag.

[2] Wolfram, S. 1986. *Theory and applications of cellular automata*. Singapore: World Scientific.

[3] Toffoli, T., and N. Margolus. 1987. *Cellular automata machines*. Cambridge, MA: MIT Press.

[4] Talia, D. 2000. Cellular processing tools for high-performance simulation. *IEEE Computer*, September, 44-52.

[5] Sipper, M. 1999. The emergence of cellular computing. *IEEE Computer*, July, 18-26.

[6] Wolfram, S. 2002. *A new kind of science*. ***CITY?***: Wolfram Media, Inc.

[7] Wainer, G., and N. Giambiasi. 2001. Application of the Cell-DEVS paradigm for cell spaces modeling and simulation. *SIMULATION* 71 (1): 22-39.

[8] Wainer, G., and N. Giambiasi. 2001. Timed Cell-DEVS: Modeling and simulation of cell spaces. In *Discrete event modeling & simulation: Enabling future technologies*, edited by ***EDITOR?***. New York: Springer-Verlag.

[9] Zeigler, B., T. Kim, and H. Praehofer. 2000. *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. New York: Academic Press.

[10] Ameghino, J., and G. Wainer. 2000. Application of the Cell-DEVS paradigm using N-CD++. In *Proceedings of the 32nd SCS Summer Computer Simulation Conference*, Vancouver, Canada.

[11] Ameghino, J., A. Troccoli, and G. Wainer. 2001. Modeling and simulation of complex physical systems using Cell-DEVS. In *Proceedings of 34th IEEE/SCS Annual Simulation Symposium*, Seattle, WA.

[12] Troccoli, A., J. Ameghino, F. Iñon, and G. Wainer. 2002. A flow injection model using Cell-DEVS. In *Proceedings of the 35th IEEE/SCS Annual Simulation Symposium*, San Diego, CA.

[13] Giambiasi, N., B. Escudé, and S. Ghosh. 2000. GDEVS: A Generalized Discrete Event Specification for accurate modeling of dynamic systems. *Transactions of the SCS* 17 (3): 120-34.

[14] Giambiasi, N. 1998. Abstractions à événements Discrets de Systèmes Dynamiques. *Journal Européen des Systèmes Automatisés (RAIRO APII, Automatique-Productique Informatique Industrielle–Hermes)* 32 (3): 275-311.

[15] Naamane, A., N. Giambiasi, and D. Bakop. 1998. A new approach for discrete event simulations. *IEE Electronics Letters* 34 (16): 1615-6.

[16] Wainer, G. 2002. CD++: A toolkit to define discrete-event models. *Software, Practice and Experience* 32 (3): 1261-1306.

[17] Goldschlager, N., and M. Goldman. 1989. *Principles of clinical electrocardiography*. Norwalk, CT: Appleton & Lange.

[18] Alberts, B., D. Bray, L. Lewis, M. Raff, K. Roberts, and D. Watson. 1983. *Molecular biology of the cell*. New York: Garland.

[19] Hodgkin, A., and A. Huxley. 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology* 117:500-44.

[20] Saxberg, B., and R. Cohen. 1991. *Theory of heart*. New York: Springer Verlag.

[21] Aschenbrenner, T. 2000. The hybrid method: Simulation of the electrical activity of the human heart. *International Journal of Bioelectromagnetism* 2 (2): ***000-000***.

[22] Fenton, F. 2000. Numerical simulations of cardiac dynamics: What can we learn from simple and complex models? *IEEE Computers in Cardiology* 27:251-4.

[23] de San Miguel, L., and G. Wainer. 2001. Implementing the Hodgkin-Huxley model in CD++. Internal report, Universidad de Buenos Aires, Argentina.

[24] Zeigler, B. P. 1998. DEVS theory of quantization. DARPA Contract N6133997K-0007, ECE Department, University of Arizona, Tucson.

[25] Wainer, G., and B. Zeigler. 2000. Experimental results of timed Cell-DEVS quantization. In *Proceedings of AIS'2000*, Tucson, AZ.

Gabriel A. Wainer *received M.Sc. (1993) and Ph.D. degrees (1998, with highest honors) at the Universidad de Buenos Aires, Argentina, and Université d'Aix-Marseille III, France. He is an assistant professor in the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). He was an assistant professor in the Computer Sciences Department at the Universidad de Buenos Aires, Argentina, and a visiting research scholar at the Arizona Center of Integrated Modeling and Simulation (ACIMS, University of Arizona) and LSIS, CNRS, France. He has published more than 100 articles in the field of operating systems, real- time systems, and discrete event simulation. He is author of a book on real-time systems and another on discrete event simulation. He is an associate editor of the* Transactions of the SCS *and the* International Journal of Simulation and Process Modeling *(*Inderscience*). He is a senior member of the Board of Directors of the Society for Computer Simulation International (SCS). He is also a coassociate director of the Ottawa Center of the McLeod Institute of Simulation Sciences and a coordinator of an international group on DEVS standardization. He has received numerous grants, scholarships, and awards, including Carleton University's Research Achievement Award (2005-2006).*

Norbert Giambiasi *has been a full professor at the University of Aix-Marseille since 1981. In October 1987, he created a new engineering school and a research laboratory, LERI, in Nîmes, located in southern France. He was the director of research and development of this engineering school. In 1994, he returned to the University of Marseilles, where he created a new research team in simulation. He has written a book on CAD and is the author of more than 150 articles published internationally. He has continued to be the scientific manager of more than 50 research contracts with E.S. Dassault, Thomson, Bull, Siemens, CNET, Esprit, Eureka, Usinor, and others and has supervised more than 40 Ph.D. students. He has been a referee for both national and European research projects. He also referees many articles for SCS publications. His current research interests converge on specification formalisms of hybrid models, discrete event simulation of hybrid systems, CAD systems, and design automation. He is the director of the Laboratory of Information and Systems Sciences, a new laboratory integrated by more than 100 researchers.*