CELL-DEVS MODELS WITH TRANSPORT AND INERTIAL DELAYS

Gabriel A. Wainer § *
gabrielw@dc.uba.ar

Norbert Giambiasi §
Norbert.Giambiasi@iuspim.u-3mrs.fr

§ DIAM-IUSPIM
Université d'Aix-Marseille III
Av. Escadrille Normandie Niemen
13397 Marseilles Cédex 20 - FRANCE

* Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Pabellón I - Ciudad Universitaria
Buenos Aires (1428) - ARGENTINA

KEYWORDS

Object-oriented modelling; Discrete simulation; grid models.

ABSTRACT

In this work new paradigms for complex modeling are presented. The new formalisms are based on the DEVS and Cellular Automata formalisms. To reduce the complexity of the models, Inertial Delays and Transport Delays constructions are also included. The combination of these modeling techniques makes available a set of tools to model complex problems. The utility of the formalisms is shown through a simple model used to simulate the behavior of a complex real system.

INTRODUCTION

Computer simulation techniques are widely used to study complex systems. In this way, they help researchers and designers to understand the dynamic behavior of real systems where analytic solutions are impossible to find. Several modeling techniques are independent of the simulators, improving the software development through validation of the models.

One of such formalisms, known as DEVS (Discrete Events Systems Specification) is based on discrete events, and was proposed by Zeigler (Zeigler 1976, Zeigler 1990). It allows modular description of the phenomena to model, and attacks the complexity using a hierarchical approach. This hierarchical modeling strategy allows the reuse of created and tested models, enhancing the security of the simulations, reducing the testing time and improving productivity.

Cellular Automata (CA) formalism is also well suited to describe complex systems with different description levels (Wolfram 1986). A conceptual CA is an infinite regular n-dimensional lattice. Each cell state is discrete and is modified in discrete time steps. To do so, it uses a local transition function based on the present cell state and a finite set of nearby cells (called the cell's neighborhood).

This formalism uses discrete time basis, posing restrictions in the precision of the model. In the particular case of complex CA, large amounts of compute time will be required. Besides, for several cases, at each time step most cells of the automaton will not need actualization. These problems does not exist in asynchronous automata, that evolve in continuous time. The use of a continuous time base allows to achieve higher time precision, periods of inactivity are skipped, and the use of the computer resources is improved.

Transport and Inertial Delays are two basic constructions usually employed in the circuit design domain. Transport Delays reflect the straightforward propagation of signals over lines of infinite bandwidth. Inertial Delays, instead, allow to define preemptive semantics in the models. These constructions also can be used to model other physical phenomena that can be described as cell spaces (i.e., fire propagation in a wood, behavior of fishes in the sea, urban traffic, ecological systems, etc.).

In this work, several formalisms to describe cell spaces are analyzed. The paradigms are based on the DEVS and Asynchronous CA formalisms, and are combined with transport or inertial delays constructions.

FORMAL DESCRIPTION OF DEVS CELLS MODELS

In (Wainer and Giambiasi 1997) several formal descriptions of CELL DEVS models were defined. These specifications allow to describe a cellular model as DEVS cells with multiple inputs and a transport or inertial delays for each cell. In this section the main aspects of these formalisms are recalled.

A transition delay modular Cell-DEVS atomic model is formally described as:

$$TDC = < I, X, S, Y, N, \delta int, \delta ext, d, \tau, \lambda, ta >$$

$I = <\eta, P^X, P^Y>$ defines the model interface. Here, $\eta \in \boldsymbol{N}$ is the neighborhood size, and, for I =X or I =Y, P^I is a port definition (input or output respectively), where P^I ={ (N_i^I, T_i^I) / $\forall$ i $\in$ [1, η], $N_i^I \in [I_1, I_\eta]$ (port name), and T_i^I =binary (port type)};

X = {0, 1} is the set of input external events;

S is the state set, where S ={(s, phase, σqueue, σ) / s $\in$ {0,1}, phase $\in$ {active, passive}, σqueue = { $(a_1,...,a_m)$ / m $\in \boldsymbol{N}$ and $\forall$ i $\in$ [1,m], i $\in \boldsymbol{N}$, $a_i \in \boldsymbol{R}_0^+ \cup \infty$}, and $\sigma \in \boldsymbol{R}_0^+ \cup \infty$} ;

Y = {0, 1} is the set of output external events;

$N \in \{0, 1\}^{\eta}$, is the set of the neighborhood's binary states;

$\delta int: S \rightarrow S$ is the internal transition function;

$\delta ext: QxX \rightarrow S$ is the external transition function, where Q is the state set defined by $Q = \{ (s, e) / s \in S$, and $e \in [0, ta(s)]\}$;

$d \in \boldsymbol{R}_0^+$ is the transport delay for the cell;

$\tau: SxN \rightarrow S$ is the local transition function;

$\lambda: S \rightarrow Y$ is the output function; and

$ta: S \rightarrow \boldsymbol{R}_0^+ \cup \infty$, is the time advance function.

The formal specification of a cell with inertial delays only changes the state definition:

$S = \{ (s, phase, f, \sigma) / s \in \{0,1\}$, phase $\in$ {active, passive}, $f \in \{0,1\}$, and $\sigma \in \boldsymbol{R}_0^+ \cup \infty \}$;

Variable f states for the "feasible" future state for the cell. That is, if the input is kept during the inertial delay, the future state will be f. Another change is in the semantic of variable d that now represents the value for the inertial delay.

```
δext((s, phase, squeue, s), e, x) = (s', phase, squeue, s), where
        s = s'
        if (phase = passive) then {
                phase = active;
                σ = d; } /* Transport Delay */
        else {  for (ai ∈ squeue) ai = ai - e;
                σ = σ - e; }
        σqueue = insert(σqueue, d);

δint(s, phase, σqueue, σ) = (s', phase, σqueue, σ), where
        s' = τ(s, N);
        if (s != s') changed = TRUE;
        else         changed = FALSE;

        for (ai ∈ σqueue) ai = ai - first(σqueue);
        σqueue = tail (σqueue);

        if empty(σqueue) {
                phase = passive;
                σ = ∞; }
        else {
                phase = active;
                σ = first(σqueue); }
```

Figure 1. Definition of transition functions.

The transport delay model allow to introduce a delay between the occurrence of an external transition function and the state change of the cell. Only when the transport delay is consumed, the internal transition function is executed and the system changes its state. The σqueue is introduced because new external events can occur while the transport delay is consumed. These must recorded and later executed by the internal transition functions. The behavior of both transition functions is similar of those defined in (Ghosh and Giambiasi 1996).

In this definition, insert, first, tail and empty are the traditional functions employed to manage a FIFO queue. The external transition function schedules a new time for an internal transition function. To do so, it uses the value of the transport delay, that is queued in the σqueue. The local transition function is started when σ=0. The λ function will only execute if the status has changed. Therefore, the state change is recorded in a flag variable that can be accessed by the λ function, that activates only if there was a change.

The Inertial Delay constructions allow to represent a behavior with preemptive semantic. The construction says that, if an input value is not kept a certain period (the inertial delay), the state change is not recorded. Instead, if the value is kept during that time, the state changes after the delay.

To model this kind of delays, the transition functions have been redefined.

```
δext((s, phase, σqueue, σ), e, x) = (s', phase, σqueue, σ), where
        s' = s;
        if (phase = passive) then {
                phase = active;
                σ = d; }  /* Inertial Delay */
        else
                { σ = σ - e;
                if (σ > 0 .AND. f != x) σ = d;
                 /* Preemption */ }
        f = x;

δint(s, phase, σqueue, σ) = (s', phase, σqueue, σ), where
        s' = τ(f, N);
        if (s != s') changed = TRUE;
        else         changed = FALSE;

        if (e = 0) { phase = passive;
                σ = ∞; }
```

Figure 2. External and internal transition function for inertial delays models.

The last arrived event can be preempted if a new external event (with different value) arrives before the end of the inertial delay. If a new external event has the same value of the old one, the result is equivalent to have a unique external event.

The atomic cell models can be coupled with other cell models, forming a multicomponent cell model. Any of these models can be also combined with DEVS models, allowing the definition of complex hierarchical models. The coupled cell models are defined as:

GCTD = < I, X, Y, Xlist, Ylist, η ,N, {m, n}, C, B, Z, Zij, select >

$I = <\eta, P^x, P^y>$ represents the definition of the modular model interface, defined as in atomic models, but considering $P^I = \{ (N(f,g)^I, T(f,g)^I) / \forall (f,g) \in Xlist, N(f,g)^I = I(f,g)_k$ (port name), and $T(f,g)^I$ = binary (port type)};

X = {0, 1} is the set of input external events;

Y = {0, 1} is the set of output external events;

Ylist = { (k,l) / k ∈ [0,m], l ∈ [0,n]} is the list of output coupling;

Xlist = { (k,l) / k ∈ [0,m], l ∈ [0,n]} is the list of input coupling;

$\eta \in \boldsymbol{N}$ is the neighborhood size;

N is the neighborhood set, defined as $N = \{ (i_p,j_p) / \forall p \in \boldsymbol{N}, p \in [1,\eta] => i_p, j_p \in \boldsymbol{Z} \wedge i_p, j_p \in [-1, 1] \}$;

$\{m, n\} \in \boldsymbol{N}$ is the size of the cell space;

C is the cell space set, defined as $C = \{C_{ij} / i \in [1,m], j \in [1,n]\}$, where

$$C_{ij} = < I_{ij}, X_{ij}, S_{ij}, Y_{ij}, \delta_{ij}, d_{ij}, \lambda_{ij}, \tau_{ij}, ta_{ij}>$$

is a cell-DEVS component;

B is the border cells set, where $B = \{\varnothing\} \cup \{c_{ij} / \forall i = 1 \vee i = m \vee j = 1 \vee j = n, c_{ij} \in C_{ij}$ is a self-generating state cell};

Z is the translation function, defined by:

$Z: P_{ij}^{Yq} \rightarrow P_{kl}^{Xq}$, where $P_{ij}^{Yq} \in I_{ij}$, $P_{kl}^{Xq} \in I_{kl}$, $q \in [0, \eta]$ and, $\forall (f, g) \in N$, $k = (i+f) \bmod m$; $l=(j+g) \bmod n$;
$Zij: Y(f,g)_i \rightarrow X(k,l)_j \forall (f,g) Ylist_i$, and $(k,l) Xlist_j$.; and

select, is the tie-breaking selector function, with the restriction that $select \subseteq mxn \rightarrow mxn / \forall E \neq \{\varnothing\}$, $select(E) \in E$.

It can be see that the models only allows binary states, but the definition can be extended by considering X and Y (and the corresponding I/O ports) as sets in ***Z*** or ***R***. The transition functions should compute their results using any of these domains.

If an event occurs in one cell, the neighbors are influenced through the execution of the Z function. Besides, certain cells in the space are chosen as input and output cells, and will be included in the Xlist and the Ylist respectively. Xlist is a list of cell's positions where the inputs to the model are received. Ylist records the cells whose outputs will be sent to the other models in the hierarchy.

When a cell-DEVS model is executed, the Zij function translates inputs into outputs by using both lists. The names of the input and output ports are also defined by using the contents of the Xlist and Ylist.

The specification models here presented are independent of the simulation technique used. Therefore, they allow to specify the system behavior independently of the implementation details of the chosen simulation technique. In the following section an example will show the application of the formalism to model a complex system.

AN APPLICATION EXAMPLE

In this section the main aspects presented in the previous section are shown through an example of application. Coupled cell models can be mixed with other basic models in a DEVS hierarchy. In this way complex models consisting of several submodels with different behavior can be built using different paradigms or abstraction levels.

In this case, several models are integrated to simulate the complex behavior of the traffic in a section of urban population.

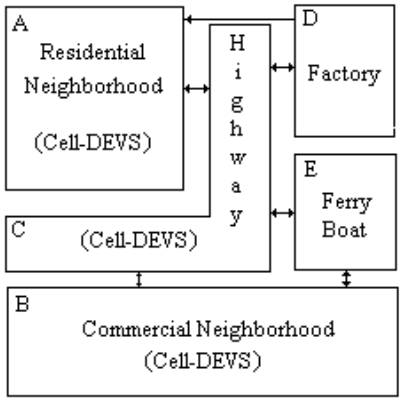

Figure 3. Coupling different DEVS models

Model A is a Cell-DEVS model representing particles of smoke on the air of a residential neighborhood. It is used to study pollution (depending on influences of the traffic in the highway, and smoke of the factory). Model B represents the traffic movement in a commercial neighborhood, to study the traffic flow (i.e., to improve semaphore synchronization). Model C represents a one-way highway passing across neighbors A and B (to study the traffic flow in the highway). Atomic model D represents a factory: trucks arrive from the highway, and new trucks filled with products get into the highway. It is used to schedule the flow of trucks to the factory. Finally, atomic model E represents the entrance to a ferry-boat that connects the city with an island. It is used to determine the optimal number of boats depending on the hour.

To build the simulation, the behavior of each submodel must be defined. We will only concentrate in the definition of the cell models (behavior of traditional models such as D or E will be skipped).

The first step to build the simulator for a cell space is to define the behavior for each cell in the model. The basic behavior is defined by the local transition functions. To do so, a simple declarative specification language has been defined (Wainer and Giambiasi 1997). The language only allows bi-

nary or three-states, and a compiler translates the specification to the internal behavior representation of the cell-DEVS models.

Each sentence is a boolean expression including relational operations and calls to predefined functions (including transport and inertial delay functions). The results for each cell are computed by analyzing each sentence, and considering the present state for the cell and its neighborhood. In the following figure the behavior for each cell is shown. The aij variables define a 9x9 matrix representing the cell neighborhood. The neighborhood state is taken as precondition for the local transition function. The result is the new state for the neighborhood.
Model A (Residential Neighborhood)

Result	Neighborhood state
aij:=1 in_delay(wind)	a22 = 1

Model B (Commercial Neighborhood)

Result	Neighborhood state
a22:=1 a32:=0 tr_delay(sp.)	a22 = 0 AND a32 = 1 /* Normal flow - Northbound */
a22:=1 a23:=0 tr_delay(sp.)	a22 = 0 AND a32 = 1 /* Normal flow - Westbound */
a22:=1 a23:=0 tr_delay(sp.)	a22 = 0 AND a23 = 1 /* Crossing the car on the right passes */

Model C (Highway)

Result	Neighborhood state
a22:=1 a32:=0 tr_delay(sp.)	a22 = 0 AND a32 = 1 /* Normal flow */
a22 := 1 a33 := 0 tr_delay(sp.)	a22 = 0 AND (a32 = 0 AND a33 = 1 AND a23 = 1) /* Surpassing by the left */
a22 := 1 a31 := 0 tr_delay(sp.)	a22 = 0 AND (a32 = 0 AND a31 = 1 AND a21 = 1) /* Surpassing by the right */

Figure 4 - Specification of local transition rules

The example has been simplified by adding several constraints (although the rules can be easily extended to represent more complex phenomena). First, it is supposed that the highway is one-way. The commercial neighborhood also has one-way streets (Northbound or Westbound), and no semaphores are modeled. In the pollution model, if a particle stays in a cell for certain time, it is diffused to the neighbors. The transport delays allow to model the acceleration delay of the cars. The inertial delay has been used to model the pollution diffusion (if a particle is removed by wind, pollution does not expand).

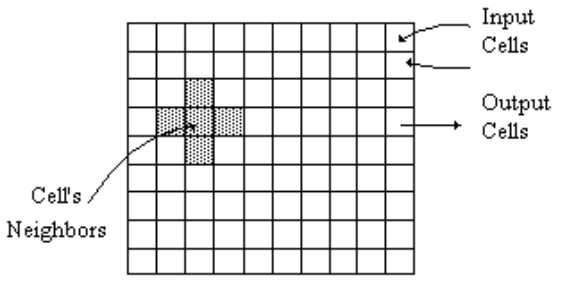

Figure 5 - Definition of cell space for model A, and input/output cells of the model.

After the definition of the cell behavior, X and Y lists should be defined for each model. Let us analyze the complete definition of the cell space A (supposing that the components are numbered in ascending order using their letter name).

According with the previous definition for the interface and port names,

$I = <5, P^X, P^Y>$, where P^X={X(1,15), X(2,15) };
P^Y = { Y(5,15) };

m = 9; n = 10;

N = { (0,0}, (-1,0), (1, 0), (0,1), (0,-1) };

B = {∅};

C is the cell space set, defined as $C = \{C_{ij} / i \in [1,9], j \in [1,10]\}$, where each C_{ij} is the cell-DEVS pollution component;

Z function is defined by the cells' neighborhood.

Let us suppose that $Xlist_2$ = { (20,5) } is the cell in model B where cars from the neighborhood enters into the highway. $Ylist_2$ = {(1,1)} is the connection from the neighborhood into this entrance to the highway. Let us suppose that cell (1,15) receives pollution from the factory, and cell (2,15) receives similar values from the highway. These values are diffused through the cell space to simulate the pollution of each block in the neighborhood.

Hence, $Xlist_1$ = { (1,15); (2,15) }; $Ylist_1$ = { (5, 15) }. Therefore, the Zij function for this model will be defined as:

Z12: $Y(5,15)_1 \rightarrow X(20,5)_3$
Z21: $Y_2 \rightarrow X(1,15)_1$
$Y(1,1)_2 \rightarrow X(2,15)_1$

Output port Y(5,15) should be connected with the input port X(20,5). An output port in model D should be connected with X(1,15). Output port Y(1,1) of model C should be connected with input port X(2,15).

To complete the specification of the model, the user should define the behavior of the border cells. To do so, the specification language also can be used. Finally, the priorities to treat simultaneous events should be specified. These are used to classify the imminent cells in the cell space. It is used to define the *Select* function of the DEVS specification (or the *dcon* function if the R-DEVS (Chow and Zeigler 1994) formalism is used).

Initial conditions for the cells and conditions to stop the simulation cycle be defined. With these parameters, an array of atomic cell objects is created, the influencees for each cell are defined, and the simulation can proceed (Wainer et al. 1997).

CONCLUSION

This work was devoted to the presentation of paradigms for cellular DEVS models. Inertial Delays and Transport Delays constructions are also included, allowing to easily define complex behavior of real systems.

The combination of these modeling techniques allows to build tools to model complex problems. The specifications allows the automatic definition of a complete DEVS model with different submodels. These specifications are completely independent of the simulators, and can be used to validate the correctness of the model.

The use of a discrete events formalism as a basis improves the precision of the models, and also reduces the execution times of the simulations. At present, a simulation environment based on these formalisms is being implemented [Wai97b], and performance improvements have been considered based on parallel implementations of the simulators.

REFERENCES

Chow, A. and Zeigler, B. 1994. "Revised DEVS: a parallel, hierarchical, modular modeling formalism". *Proceedings of the. Winter Simulation Conference.*

Ghosh, S. and Giambiasi, N. 1996."On the need for consistency between the VHDL language constructions and the underlying hardware design". *SCS* .... pp. 562-567.

Wainer, G. and Giambiasi, N. 1997. "Modeling and simulation of Cell-DEVS models". Technical Report TR-97-004, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires. Argentina.

Wainer, G.; Frydman, C.; Giambiasi, N. 1997. "An environment to simulate cellular DEVS models". *Proceedings of the SCS European Multiconference on Simulation.* Istanbul, Turkey.

Wolfram, S. 1986. *Theory and applications of cellular automata*. Vol. 1, Advances Series on Complex Systems. World Scientific, Singapore.

Zeigler, B. 1976. *Theory of modeling and simulation*. Wiley.

Zeigler, B. 1990. "Object-oriented simulation with hierarchical modular models". Academic Press.