# N-dimensional Cell-DEVS models

Gabriel A. Wainer
Gabriel.Wainer@sce.carleton.ca

*Department of Systems and Computer Engineering*
*Carleton University*
*4456 Mackenzie Building*
*1125 Colonel By Drive*
*Ottawa, ON. K1S 5B6. Canada.*

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

*LSIS*
*Université d'Aix-Marseille III*
*Av. Escadrille Normandie Niemen*
*13397 Marseilles*
*Cédex 20 - FRANCE*

**Abstract**

This article presents an extension to the timed binary Cell-DEVS paradigm. The goal is to allow the modelling of n-dimensional generic cell spaces, including transport or inertial delays for each cell. The automatic definition of cell spaces is achieved, simplifying the construction of new models. The model definition is independent of the simulation mechanism, easing the verification of the structural models. It was shown that the Cell-DEVS models can be integrated in a DEVS hierarchy, improving the definition and description of complex systems. This approach allows improvements in the execution times and precision for the cell spaces simulations due to the use of a continuous time base.

Keywords: DEVS models, Modelling Paradigms, Cellular Automata, Discrete event simulation, Cell-DEVS models.

## 1. INTRODUCTION

Complex physical systems have been studied for centuries using different approaches. In most cases, partial differential equations have been the tool of choice. Nevertheless, the appearance of digital computers allowed to attack the problem from different point of views. Even in these days, difference equations are implemented in digital computers to solve this kind of problems. Unfortunately, the complexity of certain problems is such that no solutions can be found. In these cases, the use of computer simulation allowed attacking more complex problems, providing solutions to specific problems.

Many of these systems can be represented as cell spaces. The Cellular Automata formalism (Wolfram 1986, Toffoli and Margolus 1987) has been widely used to describe complex systems with these characteristics. Cellular Automata are discrete-time discrete models described as cells organized as n-dimensional infinite lattices. These automata evolve by executing a global transition function that updates the state of every cell in the space. Each cell in the automaton has a discrete value that is changed by a local computation function. The behavior of this function depends on the results of a function that executes locally in each cell. This function uses the present value for the cell and a finite set of neighbor cells to compute the new state.

Conceptually, these local functions are computed synchronously and in parallel, using the state values of the present cell and its neighbors. This discrete time paradigm constrains the precision and efficiency of the simulated models. The use of discrete time poses constraints in the precision and execution performance of these complex models. To achieve the desired accuracy, smaller time slots must be used, producing higher needs of processing time. To avoid these problems, asynchronous solutions can be used.

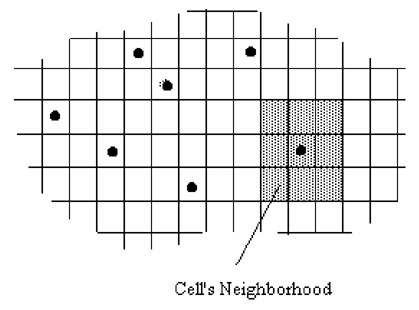

***Figure 1. Sketch of a Cellular Automaton***

Furthermore, it is usual that several cells do not need to be updated in every step, wasting computation time. These problems can be solved using a continuous time base, providing instantaneous events that can occur asynchronously at unpredictable times. This approach was considered in (Zeigler 1976, Zeigler 1984), where discrete event cellular models were presented. Discrete event cellular models were applied in real world applications in later works (Moon et al. 1996, Zeigler et al. 1998). These works presented the use of DEVS (Zeigler 1976, Zeigler et al. 2000) as the modelling technique to be applied to improve the performance in cellular models.

DEVS is used to specify formally discrete events systems using a modular description. The quantitative complexity of the problems is attacked by using a hierarchical approach. A model is seen as composed by behavioral (atomic) submodels than 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 integrated hierarchically. This strategy allows the reuse of tested models, improving the safety of the simulations and allowing to reduce the development times. DEVS provides the advantages of being a formal approach. Formal specification mechanisms are useful to improve the security and development costs of a simulation. A formal conceptual model can be validated, improving the error detection process and reducing testing time. DEVS models are closed under coupling, therefore, a coupled model is equivalent to an atomic one, improving reuse. DEVS supplies facilities to translate the formal specifications into executable models. In this way, the behavior of a conceptual model can be validated against the real system, and the response of the executable model can be verified against the conceptual specification.

DEVS, as a discrete event paradigm, uses a continuous time base, which allows accurate timing representation. Precision of the conceptual models can be improved, and CPU time requirements reduced. Higher timing precision can be obtained without using small discrete time segments (that would increase the number of simulation cycles).

Recalling the definitions, a DEVS atomic model can be formally described as:

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

where

***X*** is the input events set;

**S** is the state set;

***Y*** is the output events set;

$d_{\mathbf{int}}$: $S \rightarrow S$, is the internal transition function;

$d_{\mathbf{ext}}$: $Q \times X \rightarrow S$, is the external transition function; where $Q = \{ (s, e) / s \in S, \text{and } e \in [0, D(s)]\}$;

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

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

Models use input/output ports to communicate, which defines the model's interface. Each state in a model has a given lifetime, defined by the duration function. Once the lifetime of a given state finishes, the internal transition function is activated to produce an internal state change. Before this change, the present state of the model can be spread through the output ports. These ports allow events to be sent to other models. The values are sent by the output function, which must execute before activating the internal transition. 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. The external transition function computes a new state for the model using the present state, the input values, and the elapsed time for the model (defined by the duration function). Every time a transition function is activated, a new lifetime must be associated with the new state.

An atomic model can be integrated with other DEVS models to build a structural model. These models are called coupled, and are integrated by base models, that is, atomic or other coupled ones. DEVS coupled models are formally defined as:

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

where

***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

$\forall$ i $\in$ D, $\mathbf{M_i}$ is a basic DEVS model, where

$$M_i = < X_i, S_i, Y_i, \delta_{inti}, \delta_{exti}, ta_i >$$

$\mathbf{I_i}$ is the set of influencees of model i, and $\forall$ j $\in$ $I_i$, and

$\mathbf{Z_{ij}}$: $Y_i \rightarrow X_j$ is the i to j translation function.

Finally, **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.

In (Wainer and Giambiasi, 2001) the Timed Cell-DEVS formalism was presented as a combination of the DEVS and Cellular Automata with timing delays. This approach allows describing cell spaces as discrete events models, where each cell is seen as a DEVS atomic model that can be delayed using several constructions (Giambiasi and Miara, 1976). The paradigm included binary or three-state values for each of the cells in the space, and only allowed two-dimensional models. This work is devoted to present an extension to a more general formalism that allows to model n-dimensional spaces with generic state sets. These specifications allow to define complex models, defining different behavior in each dimension and combining them without needing extra coupling information.

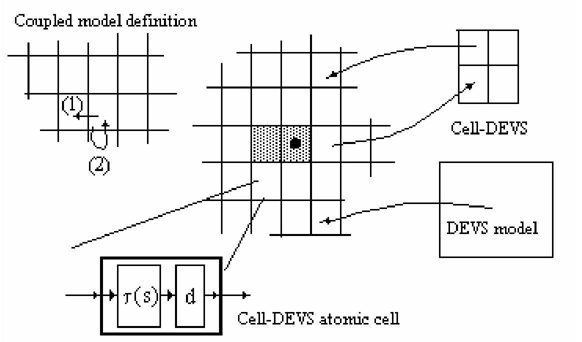

***Figure 2. Informal definition of Cell-DEVS (Wainer 2000).***

In the Timed Cell-DEVS formalism, each cell is defined as an atomic model, and a procedure to couple cells is depicted. Inertial and Transport delays allow to define complex behaviors for each cell, improving the definition for each of the submodels. We introduced two kind of delays with different semantics to allow the construction of models at two levels of accuracy. Transport delay has an anticipatory semantics, that is to say that every input event is just delayed. This is an extension of discrete event models with implicit time representation in which event are only ordered. Inertial delays allow representing more complex temporal behavior because they have preemptive semantics. An scheduled event will not necessary be executed. For example, this can of delay allows to analyze frequency responses of systems (Giambiasi and Ghosh, 1996). Delay constructions were adapted and included as a functional component of each cell. The extensions allow to define explicit timing for each cell, providing a simple mechanism to define it. Each cell is now built as a local computing function combined with a delay construction, avoiding other definition mechanisms.

These constructions are useful to represent different phenomena. For instance, the transport delay can be used to represent the fire spread rate in a wood fire simulation (the inverse of the delay length). In addition, an inertial delay could be used to represent the activity of firefighters or rain in the area. Let us suppose that a cell is set on fire when any of its neighbors is on fire. The delay is used to represent the time taken by the fire to burn the cell and spread to the close cells. In this case, a state change with inertial delay is scheduled, representing a cell non burning that is set on fire. If the delay is consumed, the event is transmitted and the crossing receives fire. Instead, let us suppose that in the meantime, an external event, representing firefighter action is received. The neighborhood of the left cell has changed, and the local function must be computed. As the previous state of the cell has changed, the scheduled event representing fire spread is preempted.

The following sections will be devoted to present the definitions for the formalism, considering n-dimensional spaces and showing the equivalence with DEVS models. First, a formal description for cellular automata is depicted. Section three presents the formal specification for the Cell-DEVS paradigm. Here, a definition for one cell is analyzed. After, closed cell spaces are studied. Finally, cell spaces that can be coupled with other DEVS models are presented.

## 2. FORMAL DEFINITION OF CELLULAR AUTOMATA

This section is devoted to present a formal definition for different paradigms associated with cellular automata. It includes formal descriptions for conceptual and executable cellular automata. In the latter case, synchronous and asynchronous approaches are considered. The section introduces several specifications for cellular models, which will be used in the following sections as a base for the Cell-DEVS definitions. We include several useful notations, and present semantics definition for these cellular models, that can be used as a comparison with the definitions in following sections.

### 2.1 Conceptual Cellular Automata

A conceptual cellular automaton can be defined as:

$$CCA = < S, n, C, N, T, \tau, c.\mathbf{Z}_0^+ >$$

where

**S** is the alphabet used to represent the state for each cell;

**n** is the dimension for the cell space;

**C** is the state set for the cell space;

**N** is the neighborhood set;

**T** is the global transition function;

t is the local computation function; and

$c.\mathbf{Z}_0^+$ is the discrete time base for the cellular automata.

**Notation 1**

From now, $C_c \in S$ will define the status for the cell c, being $c \in \mathbf{Z}^n$, $c = (i_1,...,i_n)$ the cell's position into the n-dimensional cell space. Here, $\forall\ k \in [1,n]$, $i_k \in \mathbf{Z}$ is the position of the cell in the k-eth dimension.

For conceptual cellular automata, $\forall\ k \in [1,n]$, $i_k \in [-\infty,\infty]$.

Using this notation, each of these sets can be defined as follows:

- $\mathbf{S} \subseteq \mathbf{Z} \wedge \#S < \infty$.
- $\mathbf{n} \in \mathbf{N}$.
- $\mathbf{C} = \{ C_c / c \in \mathbf{Z}^n \wedge C_c \in S \}$.
- If the neighborhood is homogeneous, $\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall (k \in N, k \in [1, \eta]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \mathbf{Z} \}$. The $\eta$ value represents the neighborhood's size, and in this case, $\eta \in \mathbf{N} \wedge \eta = \#N$. N is usually defined as a set of adjacent cells; that is, each $v_{ki} \in [-1, 1]$. Instead, if general neighborhoods are considered, then, $\mathbf{N} = \{N_c / c \in \mathbf{Z}^n \}$, with $N_c = \{ (v_{k1},...,v_{kn})_c / \forall (k \in N, k \in [1, \eta_c]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \mathbf{Z} \}$. Here, $\eta_c \in N \wedge \eta_c = \#N_c$.
- $\mathbf{T}$: $C \times c.\mathbf{Z}_0^+ \rightarrow C$.
- t : $C_c \times N \times c.\mathbf{Z}_0^+ \rightarrow C_c$. Here, if the neighborhood is homogeneous, $C_c[t+c] = \tau(C_{c+v1}[t],..., C_{c+v\eta}[t])$, where $t \in c.\mathbf{Z}_0^+ \wedge \forall (k \in N, k \in [1, \eta])$, $vk \in N \wedge c + vk = (i_1+v_{k1},..., i_n+v_{kn})$. Instead, if non-homogeneous cellular automata are defined,
  t : $C_c \times N_c \times c.\mathbf{Z}_0^+ \rightarrow C_c$. Here, $C_c[t+c] = \tau(C_{c+v1}[t],..., C_{c+v\eta c}[t])$, where $t \in c.\mathbf{Z}_0^+ \wedge \forall (k \in N, k \in [1, \eta_c])$, $vk \in N_c \wedge c + vk = (i_1+v_{k1}, ..., i_n+v_{kn})$.
- Finally, $c.\mathbf{Z}_0^+ = \{ i / i \in \mathbf{N}, i = c.j \wedge j \in \mathbf{N} \} = \{ 0, c, 2c, 3c, ...\}$.

As it can be seen in the definition, the model is composed by an **n-**dimensional cell space (**C**). This state space progresses in discrete time steps: the time base is defined by $c.\boldsymbol{Z}_0^+$ (a set of integer values separated by a time constant). The state for each cell in the space can take a value from a finite alphabet (**S**). Several extensions of cellular automata consider continuous values for the states. In this cases, $\mathbf{S} \subseteq \boldsymbol{R}$.

The cell's neighborhood is defined as a list of h n-dimensional neighbors. In the homogeneous case, the neighbors are defined as an n-tuple of positions relative to the origin cell. This definition uses an index (k) that allows to identify the neighbor number, and a second index (i) indicating the dimension for each of the neighbors' positions. The non-homogeneous neighborhoods are defined with an array of neighborhood lists. In this case, each cell will have a neighbor's list composed by $\eta_c$ elements, which are constituted by tuples of indexes relative to the origin cell.

The state space of the automata evolves by executing a global transition function (**T**) that changes the state of the cell space. The behavior of this function responds to the execution results of local transition functions (t) that execute locally in the neighborhood for the cell (**N**). Conceptually, the computation for these local functions is done synchronously and in parallel for every cell in the space. The semantics of this behavior can be defined by the following rule:

$$\frac{C_c \in C \quad \forall\, c \in \boldsymbol{Z}^n, \qquad t \in c.\boldsymbol{Z}_0^+}{C[t+c] = T(C[t]), \quad \text{with } C_c[t+c] = \tau(N_c, C_c[t])\ \forall\, c \in \boldsymbol{Z}^n; \qquad t = t + c}$$

This definition considers that the global transition function analyzes all the cell space at the instant *t*, and then it produces a change in the cell space for the next step. The period for this step is of *c* time units. This change can be seen as the individual computation of the local transition function for each cell in the space.

## 2.2 Synchronous executable cellular automata

The previous case considered that the index for the cell space can include an infinite number of cells. As the interest is focused into models that can run in a computer, an executable synchronous cellular automata can be defined as:

$$CA = < S, n, \{t_1,...,t_n\}, C, N, B, T, \tau, c.\boldsymbol{Z}_0^+ >$$

where all the elements in the tuple represent the same sets of the previous case, and the following sets were added:

$\{\mathbf{t_1,...,t_n}\}$ is the number of cells for each of the dimensions; and
**B** is the set of border cells.

Here,

- $\mathbf{S} \subseteq \boldsymbol{Z} \wedge \#S < \infty$;
- $\mathbf{n} \in \boldsymbol{N}$ ; $n < \infty$;
- $\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, with $t_k < \infty \ \forall\ k \in [1,n]$;
- $\mathbf{C} = \{ C_c \ / \ c \in \mathbf{I} \wedge C_c \in S \}$, with

$$\mathbf{I} = \{ (i_1,...,i_n) \ / \ i_k \in \boldsymbol{N} \wedge i_k \in [1, t_k] \ \forall\ k \in [1, n] \} \qquad (1)$$

- For homogeneous neighborhoods, $\mathbf{N} = \{ (v_{k1},...,v_{kn}) \ / \ \forall\ (k \in N, k \in [1, \eta]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge t_i - | v_{ki} | \geq 0 \}$. Here $\eta \in \boldsymbol{N}$, $\eta \leq \prod_{i=1}^{n} t_i \wedge \eta = \#N$. Instead, if general neighborhoods are defined, $\mathbf{N} = \{N_c \ / \ c \in \boldsymbol{N^n}, c \in [1,t1]x...x[1,t_n]\}$, with $N_c = \{ (v_{k1},...,v_{kn})_c \ / \ \forall\ (k \in N, k \in [1, \eta_c]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge t_i - | v_{ki} | \geq 0 \}$. Here, $\eta_c \in N \wedge \eta_c = \#N_c$.
- $\mathbf{B} = \{\emptyset\}$ if the cell space is "wrapped" (that is, the cells in each border are connected with the cells in the opposite one), or
  $\mathbf{B} = \{C_b / C_b \in C \text{ with } b \in \mathbf{I} \}$, with $\mathbf{I}$ defined as in (1). In this case, B has the restriction that $\tau_b \neq \tau_c = \tau \ \forall\ (C_c \notin B) \wedge (C_b \in B)$.
- $\mathbf{T}$: $C \times c.\boldsymbol{Z}_0^+ \rightarrow C$.
- $\tau: C_c \times N_c \times c.Z_0^+ \rightarrow C_c$; where $C_c[t+c] = \tau(C_{c+v1}[t],...,C_{c+v\eta}[t])$, (2)
  with $t \in c.\boldsymbol{Z}_0^+ \wedge \forall\ (k \in N, k \in [1, \eta_c]), vk \in N_c \wedge c + vk = (i_1+v_{k1} \bmod(t_1),..., i_n+v_{kn} \bmod(t_n))$ in the case that $B = \{\emptyset\}$, and $C_b[t+c] = \tau_b(C_b[t])$, $\forall\ b \in B$, with $t \in c.\boldsymbol{Z}_0^+$. In this case, $\tau_b \neq \tau_c = \tau \ \forall\ c \notin B$, and for these ones, $C_c$ is computed like in (2). If the cell space is homogeneous, then $\eta_c = \eta \wedge N_c = N$.

This definition for executable cellular automata differs in certain aspects of that of conceptual ones. The first difference is that the cell space is bounded in each of the dimensions ($\mathbf{t_1,...,t_n}$). The number of dimensions is also finite, and the cell's indexes are bounded to finite natural numbers.

Another constraint is due to the loss of homogeneity in the cell space. This is due to the existence of a finite number of cells. Therefore, it is necessary to include a set of border cells (**B**) with different behavior than the others in the cell space. All the cells in the border have different behavior that those in the rest of the automaton. When $B \neq \{\emptyset\}$, it is used to be defined as: $B = \{C_b \ / \ C_b \in C \text{ with } b \in L\}$, being $L = \{ (i_1,...,i_n) \ / \ i_j$

$= 0 \vee i_j = t_j \; \forall \; j \in [1, n] \}$, and with $\tau_b \neq \tau_c = \tau \; \forall \; c \notin L$. When wrapped models are considered, B = {∅} and the rules defined in (2) should be used.

The semantics for the transition function T is that the local transition functions in the automata are executed simultaneously in parallel, and is defined by:

$$\frac{C_c \in C \;\; \forall \; c = \{ (i_1,..., i_n) \,/\, i_j \in [1, t_j] \; \forall \; j \in [1,n] \}, \qquad t \in c.\boldsymbol{Z}_0^+}{C[t+c] = T(C[t]),\text{ with } C_c[t+c] = \tau(N_c,C_c[t]) \quad \forall \; c = \{ (i_1,..., i_n) \,/\, i_j \in N, i_j \in [1, t_j] \; \forall \; j \in N \,,\, j \in [1,n] \};\; t = t + c}$$

The meaning is similar that in the case of conceptual automata, but bounding the cell space.

## 2.3 Asynchronous executable cellular automata

Finally the formal definition for a set of asynchronous cellular automata that can be executed is defined as:

$$ACA = < S, n, \{t_1,...,t_n\}, C, N, B, Nevs, T, \tau, \boldsymbol{R}_0^+ >$$

where all the sets are defined as in the previous cases, except by the time base (that in this case is continuous), and a Next events list (**Nevs**). These sets are defined by:

- $\mathbf{S} \subseteq \boldsymbol{Z} \wedge \#S < \infty$
- $\mathbf{n} \in \boldsymbol{N}$, $n < \infty$.
- $\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, with $t_k < \infty \; \forall \; k \in [1,n]$.
- $\mathbf{C} = \{ C_c \,/\, c \in \mathbf{I} \wedge C_c \in S \}$, with **I** defined as in (1).
- For homogeneous neighborhoods, $\mathbf{N} = \{ (v_{k1},...,v_{kn}) \,/\, \forall \; (k \in N, k \in [1, \eta]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z}$ $\wedge \; t_i - | v_{ki} | \geq 0 \}$. Here $\eta \in \boldsymbol{N}$, $\eta \leq \prod_{i=1}^{n} t_i \wedge \eta = \#N$. Instead, if general neighborhoods are defined, $\mathbf{N} = \{N_c \,/\, c \in \boldsymbol{N}^{\boldsymbol{n}}, c \in [1,t1]x...x[1,t_n]\}$, with $N_c = \{ (v_{k1},...,v_{kn})_c \,/\, \forall \; (k \in N, k \in [1, \eta_c]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \; \wedge t_i - | v_{ki} | \geq 0 \}$. Here, $\eta_c \in N \; \wedge \eta_c = \#N_c$.
- **B** = {∅} if the cell space is wrapped, or

**B** = {$C_b$ / $C_b \in$ C with b$\in$ **I** }, with **I** defined as in (1). In this case, the set B is subject to the constraint that $\tau_b \neq \tau_c = \tau \ \forall$ ($C_c \notin$ B) $\wedge$ ($C_b \in$ B).

- **Nevs** = { (c, $t$) / c $\in$ **I** $\wedge$ $t \in \boldsymbol{R}_0^+$}, where c is the position of a cell in the space, **I** is defined as in (1), and $t$ is the time of the corresponding event.
- **T**: C x $\boldsymbol{R}_0^+ \rightarrow$ C.
- $\tau$: $C_c$ x $N_c$ x $R_0^+ \rightarrow C_c$; where $C_c[t_p] = \tau(C_{c+v1}[t],...,C_{c+v\eta c}[t])$, (3)

  with $t \in \boldsymbol{R}_0^+ \wedge \forall$ (k$\in$ $N$, k$\in$ [1, $\eta$]), vk$\in N_c \wedge$ c + vk = ($i_1+v_{k1}$ mod($t_1$),..., $i_n+v_{kn}$ mod($t_n$)) when B = {$\emptyset$},

  and $C_b[t_p] = \tau_b(C_b[t])$, $\forall$ b $\in$ B, with $t \in \boldsymbol{R}_0^+$. In this case, $\tau_b \neq \tau_c = \tau \ \forall$ c $\notin$ B, and for these cases, $C_c$ is computed as in (3). Here, $t_p = \min\{t_i\}_{i=1}^{n}$, with i, p$\in$ $N$ + $t_i \in \boldsymbol{R}_0^+$, and b = $c_p \vee$ c = $c_p$, with ($t_i$, $c_i$) $\in$ Nevs. In this case, $\tau_b \neq \tau_c = \tau \ \forall$ c $\notin$ **I** (1), and for these, $C_c$ is computed as in (3). If the cell space is homogeneous, then $\eta_c = \eta \wedge N_c$ = N.

As it can be seen, most of the sets and functions defined are similar than for the synchronous case. The changes are due to the existence of a continuous time base (that is, the time variable $t \in \boldsymbol{R}_0^+$). To allow the asynchronous definition, a next event list (**Nevs**) should be included to keep the information related with the next events to be treated.

In this case, the semantics for the global transition function is different from for the previous case. Here, this function means to execute only a group of non-quiescent cells called the *imminent*. The execution of this function is done simultaneously in all the imminent cells for a given simulated time. The semantics of this behavior can be specified as:

$$\frac{C_c \in C \ \ \forall\ c \in \mathbf{I}\,(1), \qquad C_p = \{\ (c_p, t_p) / (c_p, t_p) \in \text{Nevs}\ \} \ \wedge \ \ t_p = \min\{ti\}_{i=1}^{n}}{C[t_p] = T(C[t]), \ \text{ with } C_{cp}[t_p] = \tau(N_{cp}, C_{cp}[t])\ \forall\ cp = c_p / (c_p, t_p) \in C_p;\ t = t_p \ \wedge\ \text{Nevs} = \text{Nevs} \cup N_{cp}}$$

In this case, the automaton progresses using the imminent cells ($C_p$) to execute the local computation function. The simulated time is considered that of the last executed event ($t_p$). Finally, the neighbor cells are added in the Next-events list.

## 3. CELL-DEVS FORMALISM

This section is devoted to present a paradigm used to describe n-dimensional cellular models. These models can be defined as DEVS models with different delays. Inertial and transport delays are useful to model circuit applications, and to define other class of physical phenomena that can be described as cell spaces. As previously stated, the Cell-DEVS formalism was defined as a modelling paradigm for bidimensional timed cell spaces. A simulation mechanism was introduced for these models, based on a previous formal specification for ideal transport delays (Giambiasi and Ghosh, 1996).

### 3.1 Formal specification of atomic Cell-DEVS models with transport delays.

This section introduces the definition of n-dimensional Cell-DEVS modular models with **transport delays**. This construction allows to reflect the direct propagation of signals in lines of infinite bandwidth. Each cell in the model is defined as an atomic model that can be coupled with other cells to build a complete cell space.

Figure 3 presents informally the basic contents for a cell. In this atomic model, a cell has $\eta_c$ inputs where the external events arrive. When a new external event arrives, the local computation function $\tau$ is executed using the present input values. The result of this computation is delayed during **d** time units before being transmitted to the neighbor cells. To do so, an internal event is scheduled. A queue must be used to keep the values of the computations' results with their future scheduled time, due that during the delay, new external events can arrive. When the internal event time arrives, the value is transmitted through an output port. The new state will only be transmitted if it is different from the previous one, because the influenced models only must be activated under state alteration. To allow this distinction, the cell's state keeps the present and past value for the cell.

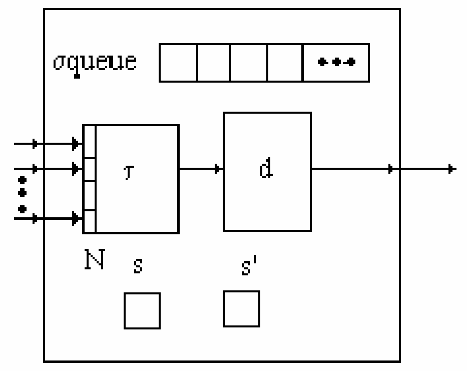

***Figure 3. Informal description of an atomic cell with transport delays.***

Hence, a Cell-DEVS atomic model with transport delays can be formally defined as:

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

where for $\#T < \infty \wedge T \in \{N, Z, \boldsymbol{R}, \{0,1\}\} \cup \{\phi\}$;

$\boldsymbol{X} \subseteq T$ is the set of external input events;

$\boldsymbol{Y} \subseteq T$ is the set of external output events;

$\boldsymbol{I} = < \eta, \mu, P^X, P^y >$ represents the definition of the model's modular interface. Here,

$\eta \in \boldsymbol{N}, \eta < \infty$ is the neighborhood's size,

$\mu \in \boldsymbol{N}, \mu < \infty$ is the number of other input/output ports, and

$\forall\ j \in [1, \eta], i \in \{X, Y\}, P_j^i$ is a definition of a port (input or output respectively), with

$P_j^i = \{ (N_j^i, T_j^i) / \forall\ j \in [1, \eta+\mu], N_j^i \in [i_1, i_{\eta+\mu}]$ (port name), y $T_j^i \in I_i$ (port type)$\}$, where

$I_i = \{ x / x \in \boldsymbol{X}$ if X $\}$ or $I_i = \{ x / x \in \boldsymbol{Y}$ if $i = Y \}$ ;

**S** $\subseteq T$ is the set of sequential states for the cell;

q is the definition of the cell's state, defined as

$\theta = \{$ (s, phase, $\sigma$queue, $\sigma$) /

s $\in$ S is the status value for the cell,

phase $\in$ {active, passive},

$\sigma queue = \{ ((v_1,\sigma_1),...,(v_m,\sigma_m)) / m \in \boldsymbol{N} \wedge m < \infty) \wedge \forall\ (i \in \boldsymbol{N}, i \in [1,m]), v_i \in S \wedge \sigma_i \in \boldsymbol{R}_0^+ \cup \infty\}$; and

$\sigma \in \boldsymbol{R}_0^+ \cup \infty$

} ;

**N** $\in S^{\eta+\mu}$ is the set of states for the input events;

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

$d_{\mathbf{int}}: \theta \rightarrow \theta$ is the internal transition function;

$d_{\mathbf{ext}}: Q x X \rightarrow \theta$ is the external transition function, where Q is the state values defined as:

$$Q = \{ (s, e) / s \in \theta \times N \times d; e \in [0, D(s)]\};$$

$t: N \rightarrow S$ is the local computation function;

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

**D**: $\theta \times N \times d \rightarrow \boldsymbol{R}_0^+ \cup \infty$, is the state's duration function.

This definition is independent of the simulation technique used. Therefore, it allows to specify the behavior of the system without considering implementation details.

Each cell has an interface, composed by a fixed number of ports used to establish the internal coupling of the cell's model. The number of these ports is equal to the size of the neighborhood for the cell. If other inputs or outputs are needed, the other defined ports are used. Each port is defined by an identifier (X for input or Y for output) and a number. Though the ports' values, input/output sets and cell's states can be defined on many different domains, the ***N***, ***Z***, ***R***, and ***Boolean*** sets were chosen. This choice was due that most applications use data in these domains, and most symbol sets can be mapped onto them. An undefined symbol ($\phi$) was included with the goal to define the state of the cell when it is not known. This allows to reflect certain situations where the present value for a cell is not known nor can be computed.

The state for each cell is defined as a set composed by its present value and phase. A queue is also used to keep the next events values and their scheduled simulated time. The **N** set is used to represent the input values for the cell and it is composed by an $\eta+\mu$-tuple ($s_1,..., s_{\eta+\mu}$), where $s_i \in S$. This set is used to record the present values used to compute the future value for the cell through the local computation function t. The duration function **D** manages the cell's lifetime. Here, **D**(s, phase, $\sigma$queue, $\sigma$, N, d) = *t* represents the time during which a cell keeps the present status if no external events are detected.

The transition and output functions (l, $d_{int}$, $d_{ext}$) have the same goals than those used in traditional DEVS models. The internal transition function is used to define state changes due to internal events, and the external transition function will express the occurrence of external events. Nevertheless, these functions are used with a specific purpose here. Each cell can have an associated delay (**d**), allowing to delay the execution of the internal transition function. The transport delay construction allows the state changes to be deferred up to the moment when the time is consumed. Also, the atomic models do not change to a passive state when the internal transition function is executed, because during the delay new external events can arrive. Therefore, a cell only passivates when the delay has been consumed and there are no more scheduled events. This behavior was presented in (Wainer and Giambiasi, 2001), and the semantics for the functions can be defined as follows:

$d_{int}$:

$$\frac{\sigma = 0; \qquad \sigma queue \neq \{\emptyset\}; \qquad phase = active}{\forall\ i \in [1, m], a_i \in \sigma queue, a_i.\sigma = a_i.\sigma - head(\sigma queue.\sigma); \qquad \sigma queue = tail(\sigma queue); \\ s = head(\sigma queue.v); \qquad \sigma = head(\sigma queue.\sigma);}$$

$$\frac{\sigma = 0; \qquad \sigma queue = \{\emptyset\}; \qquad phase = active}{\sigma = \infty; \quad phase = passive}$$

l:

$$\frac{\sigma = 0;}{out = head(\sigma queue.v);}$$

d$_{ext}$:

$$\frac{s' = \tau(N_c); \quad \sigma \neq 0; \quad e = D(\theta \times N \times d); \quad phase = active}{s \neq s' \Rightarrow (s' = s \wedge \forall\, i \in [1,m]\ a_i \in \sigma queue,\ a_i.\sigma = a_i.\sigma - e \wedge \sigma = \sigma - e;\ add(\sigma queue, <s', d>))}$$

$$\frac{s' = \tau(N_c); \quad \sigma \neq 0; \quad e = D(\theta \times N \times d); \quad phase = passive}{s \neq s' \Rightarrow (s' = s \wedge \sigma = d \wedge phase = active \wedge add(\sigma queue, <s', d>))}$$

In this case, the functions tail/head/add represent the traditional methods to manage the elements of a list. The variables σqueue, σ, phase, m, s and s' are defined by the previous specification.

## 3.2 Cell-DEVS models with inertial delays

The introduction of **inertial delays** allows to model phenomena with preemptive semantics. This construction discards the inputs of a cell if their values are not kept during a predefined period. Conversely, the input values are taken into account if the input flow is constant for this interval (called the inertial delay). The use of this construction allows to model complex behaviors in a simple fashion.

A Cell-DEVS atomic model with inertial delays can be defined as:

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

where for $\#T < \infty \wedge T \in \{N, Z, \boldsymbol{R}, \{0,1\}\} \cup \{\phi\}$;

$\boldsymbol{X} \subseteq T$ is the set of external input events;

$\boldsymbol{Y} \subseteq T$ is the set of external output events;

$\boldsymbol{I} = < \eta, \mu, P^X, P^Y >$ represents the definition of the modular model's interface. Here, $\eta \in \boldsymbol{N}$, $\eta < \infty$ is the neighborhood's size, $\mu \in \boldsymbol{N}$, $\mu < \infty$ is the number of other input/output ports, and $\forall\ j \in [1, \eta]$, $i \in \{X, Y\}$, $P_j^i$ is the definition of a port (input or output), defined as

$P_j^i = \{ (N_j^i, T_j^i) / \forall j \in [1, \eta+\mu],\ N_j^i \in [i_1, i_{\eta+\mu}]$ (port name), y $T_j^i \in I_i$ (port type)$\}$,

$I_i = \{ x / x \in \boldsymbol{X}$ if $i = X \}$ or $I_i = \{ x / x \in \boldsymbol{Y}$ if $i = Y \}$ ;

**S** $\subseteq$ T is the set of sequential states for the cell;

$q = \{ (s, phase, f, \sigma) / s \in S$, phase $\in$ {active, passive}, $f \in T$, and $\sigma \in \boldsymbol{R}_0^+ \cup \infty \}$ is the definition of the cell's state;

**N** $\in S^{\eta+\mu}$, is the set of the external input values;

**d** $\in \boldsymbol{R}_0^+$, $d < \infty$ is the inertial delay for the cell;

$d_{int}$: S $\rightarrow$ S is the internal transition function;

$d_{ext}$: $QxX \rightarrow \theta$ is the external transition function, with $Q = \{ (s, e) / s \in \theta \times N \times d \wedge e \in [0, D(s)]\}$;

t: N $\rightarrow$ S is the local computation function;

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

**D**: $\theta \times N \times d \rightarrow \boldsymbol{R}_0^+ \cup \infty$, is the duration for the cell's state.

As it can be seen, most of the sets and functions are similar to those defined for transport delayed models. The main differences can be found in the definition for the cell's state. Here, **s**, **phase** and s have the same meaning than the previous case, but **f** represents the feasible future value for the cell. If the result of the cell's computation is kept during the inertial delay, the next state for the cell will be **f**. The **d** variable defines the inertial delay, and the semantic for the transition functions must be redefined.

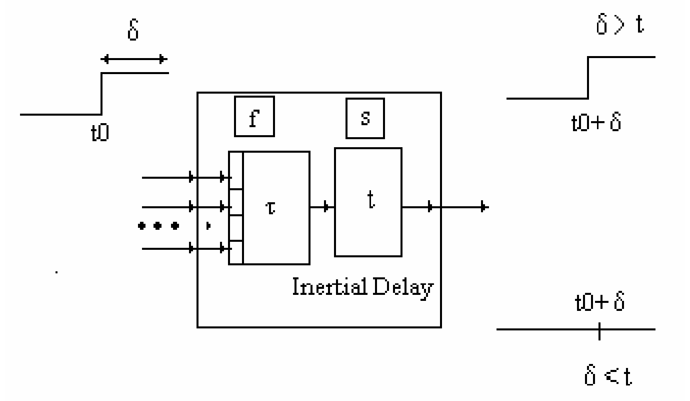

***Figure 4. Behavior of an atomic cell model with inertial delay (Wainer and Giambiasi, 2001).***

The external transition function must execute the local computation function, and keep that result as the present value for the cell. The internal transition function must detect if that input is maintained during the inertial delay. If not, the previous input is preempted. When the time specified by the delay arrives, the present value for the cell is transmitted to the neighbors. The semantic for these functions can be defined as follows:

$d_{int}$:

$$\frac{\sigma = 0; \quad \text{phase} = \text{active}}{\sigma = \infty \wedge \text{phase} = \text{passive}}$$

l:

$$\frac{\sigma = 0;}{\text{out} = s}$$

$d_{ext}$:

$$\frac{s' = \tau(N_c); \qquad \sigma \neq 0; \qquad e = D(\theta \times N \times d); \qquad \text{phase} = \text{passive}}{s \neq s' \Rightarrow ( s' = s \wedge \text{phase} = \text{active} \wedge \sigma = d \wedge f = s )}$$

$$\frac{s' = \tau(N_c); \qquad \sigma \neq 0; \qquad e = D(\theta \times N \times d); \qquad \text{phase} = \text{active}}{s \neq s' \Rightarrow s' = s \wedge (f \neq s' \Rightarrow \sigma = e \wedge \sigma = \sigma - e \wedge f = s)}$$

**Lemma 1**

The TDC and IDC models are DEVS atomic models.

**Proof:**

As previously defined, a DEVS atomic model is defined by

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

Besides, we have

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

and

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

To show the equivalence of both models, it must be determined that all the defined sets are equivalent. First, analyzing the definitions of all the models, it can be seen that $X' \equiv X$; and $Y' \equiv Y$.

Let us now define S = **T** x **T** x Sq x N' x d', with Sq = { (x, y) / x ∈ **T** ∧ y ∈ $\boldsymbol{R}_0^+ \cup \infty$}; and #Sq < ∞. Here, Sq states for the set including the possible values for σqueue, and **T** the state values for the cell (present and past, or present and feasible, depending on the case).

Now, it has to be considered that every atomic model has two defined variables: phase and σ. In this case, σ can be computed as $\sigma = \min \{yi\}_{i=1}^{\#Sq}$ / (xi, yi) ∈ Sq. Therefore, it can be said that S ≡ θ' x N' x d', and by construction of the sets, S is a well-defined set of sequential states.

In addition $\delta_{int}'$: S → S because, by definition, $\delta_{int}'$: θ' → θ', and θ' ⊆ S.

By its definition, D': θ x N x d ' → $\boldsymbol{R}_0^+ \cup \infty$, and it has been said that θ x N x d ' ≡ S.

Besides, $\delta_{ext}'$: Q x $X$ → S, because $\delta_{ext}'$: Q' x $X'$ → θ', with Q' = { (s, e) / s ∈ θ' x N' x d' ∧ e ∈ [0, D'(s)]}, and, by definition, Q' ⊆ Q ∧ $X' \equiv X$ ∧ θ' ⊆ S.

τ' is a function activated by $\delta_{ext}'$, and it can be considered as part of its behavior (as it was shown in the definitions for the behavior semantic).

Finally, λ': S → $Y$, because $Y' \equiv Y$, θ' ⊆ S, and, by definition, λ': θ' → $Y$.

Therefore, the models are equivalent, and TDC and IDC models can be considered as DEVS atomic models.

## 3.3 Formal specification of closed coupled Cell-DEVS models.

The goal of this section is to specify a formal description for coupled Cell-DEVS models. They are defined as a space of atomic cells (as the previously defined) connected by a neighborhood relationship. The present specification only will define closed models (that is, that they will not be coupled with other basic models). Therefore, as inputs and outputs are not used, it will not be necessary to define an interface for the model.

A closed coupled Cell-DEVS model can be defined as:

$$CC = < n, \{t_1,..., t_n\}, N, C, B, Z, select >$$

where

**n** is the dimension of the cell space;

{$\mathbf{t_1,...,t_n}$} is the quantity of cells in each of the dimensions;

**N** is the neighborhood set;

**C** is the cell space;

**B** is the set of border cells;

**Z** is the translation function; and

**select** is the tie-breaking function for simultaneous events.

Definition

Let us call **C** to a cell's position, where **C** = $(i_1,...,i_n)$; and $V \in \boldsymbol{Z}^n$ to the n-tuple defined as $V = (v_1,..., v_n)$, with $v_i \in \boldsymbol{Z}$, $|v_i| \leq t_i$. Therefore,

$$D = C +_t V \qquad (4)$$

will be defined as the n-tuple $D = (j_1,...,j_n)$ computed as: $j_k = (i_k + v_k) \bmod (t_k)\ \forall\ k \in [1, n]$.

**Notation 2**

The k-eth element of the neighborhood list will be called $V_k = \{ (v_{k1},...,v_{kn}) / (v_{k1},...,v_{kn}) \in N \}\ \forall\ k \in N, k \in [1, \eta]$.

**Notation 3**

$P_c^{iq}$ will define the port $P_q^i \in I_c$, with $i \in \{X, Y\}$, and $(q \in \boldsymbol{N}, q \in [1, \eta])$, where $I_c \subseteq C_c$, and $C_c$ is a TDC or IDC component.

By using these definitions, we have:

- $\mathbf{n} \in \boldsymbol{N}$, $n < \infty$;
- $\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, with $t_k < \infty\ \forall\ k \in [1,n]$;
- For homogeneous neighborhoods, $\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall\ (k \in N, k \in [1, \eta]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge t_i - |v_{ki}| \geq 0 \}$. Here $\eta \in \boldsymbol{N}$, $\eta \leq \prod_{i=1}^{n} t_i \wedge \eta = \#N$. Instead, if general neighborhoods are defined, $\mathbf{N} = \{N_c / c \in \boldsymbol{N}^n, c \in [1,t1]\text{x}...\text{x}[1,t_n]\}$, with $N_c = \{ (v_{k1},...,v_{kn})_c / \forall\ (k \in N, k \in [1, \eta_c]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \wedge t_i - |v_{ki}| \geq 0 \}$. Here, $\eta_c \in N \wedge \eta_c = \#N_c$.
- $\mathbf{C} = \{ C_c / c \in \mathbf{I} \wedge C_c = < I_c, X_c, Y_c, S_c, N_c, d_c, \delta_{int_c}, \delta_{ext_c}, \tau_c, \lambda_c, D_c >$ is a TDC or IDC component$\}$, with

$$\mathbf{I} = \{ (i_1,...,i_n) / (i_k \in N \wedge i_k \in [1, t_k])\ \forall\ k \in [1, n] \} \qquad (5)$$

- $\mathbf{B} = \{\varnothing\}$ if the cell space is "wrapped"; or

  $\mathbf{B} = \{C_b / C_b \in C \text{ with } b \in \mathbf{I} \}$, with **I** defined like in (5). In this case, the B set is subject to the constraint that $\tau_b \neq \tau_c = \tau\ \forall\ (C_c \notin B) \wedge (C_b \in B)$.

In general, B is defined as $B = \{C_b / C_b \in C \text{ with } b \in L\}$, being $L = \{ (i_1,...,i_n) / i_j = 0 \vee i_j = t_j \; \forall\, j \in [1, n] \}$, and constrained to $\tau_b \neq \tau_c = \tau \; \forall\, c \notin L$.

- **Z**: $I_C \rightarrow I_D$ is the translation function, defined (using (4)) as:

  $Z(P_C^{Y_q}) = P_D^{X_q}$, $\forall\, (q \in \boldsymbol{N}, q \in [1,\eta])$, where $\forall\, V_k \in N$, $D = C +_t V_k$; and

  $Z(P_C^{X_q}) = P_D^{Y_q}$, $\forall\, (q \in \boldsymbol{N}, q \in [1,\eta])$, where $\forall\, V_k \in N$, $D = C -_t V_k$.
- **select** = $\{ (s_1, ..., s_n) / s_i \in \boldsymbol{N}, \forall\, i \in [0, n] \}$.

In this case, the cell space **C** is a coupled model defined as an array of Cell-DEVS atomic models of fixed size ($t_1$ x ... x $t_n$). Each cell has a set of neighbors defined by the neighborhood set (**N**). This set is represented as a set of tuples of dimension n defining the relative position between the origin cell and the neighbors. These indexes cannot exceed the boundaries of the cell space.

The **B** set defines the border of the cell space, and can be of two different types. If $B = \{\emptyset\}$, every cell in the space will have the same behavior. The cells in one border will be connected with those in the opposite one using the inverse neighborhood relationship. Instead, if the border set is not empty, the cells will have a different behavior of the others in the model. The **Z** function allows to define the coupling between cells in the model. This function translates the outputs of the q-eth output port in the cell $C_c$ into values for the q-eth input port in the cell $C_D$. Finally, the **select** function defines an order of execution for the case where simultaneous events occur.

**Lemma 2**
The CC models are DEVS coupled models.

**Proof:**
A DEVS coupled model is defined as:

$$CM = < X, Y, D, \{M_d\}, \{I_d\}, \{Z_{dj}\}, select >$$

CC is a coupled model if $CC \equiv CM$, with

$$CC = < n, \{t_1,..., t_n\}, N, C, B, Z, select >$$

In this case, the following definition can be considered:

$X = \{ \emptyset \}$;

$Y = \{ \emptyset \}$;

$D \in \boldsymbol{N}$, $D \in [1, \prod_{i=1}^{n} t_i]$.

$\forall$ $d \in D$, let us consider a mapping function from $(x_1,...,x_n)$ to d, defined by

$F$: $[1,t_1]x...x[1,t_n] \rightarrow D$ in such a way that $F(c) = F(x_1,...,x_n) = d$. For instance, the function $F(x_1,...,x_n) = \sum_{j=1}^{n-1} \left[ \prod_{i=1}^{j} t_i \right] (x_{j+1} -1) + (x_1 -1)$ could be used.

For the bidimensional case (Wainer and Giambiasi, 2001), we have: $F$: $[1,f] \times [1,c] \rightarrow D$, such that $F(i,j) = d$. Here, $D \in [1, fxc]$. For instance, $F(i,j) = (i-1).c + (j-1)$.

Now, let us consider that $M_d = C_{F(c)}$ $\forall$ $c \in [1,t_1]x...x[1,t_n]$ (in the two-dimensional case, $M_d = C_{F(i,j)}$ $\forall$ $i \in [1, f]$, $j \in [1, c]$). By the definition of the mapping function, and by the Lemma 1, $M_d$ is a DEVS atomic model.

By the definition of B and the definition of $M_d$ just presented, $B \subseteq M_d \vee B = \{ \varnothing \}$.

Let us define $I_d = \{ x / x = F(c +_t V) \; \forall \; V \in N\}$, with d defined as $d = F(c)$. Then, by the definition of N, $I_d \subseteq D$. In the bidimensional case,

$$I_d = \{ x / x = F(i+n_1, j+n_2) \; \forall \; (n_1, n_2) \in N\},$$

with $d = F(i,j)$). For this case, let us verify that the expression is valid for one of the extremes of the cell space. Let $i = f-1$, $j = c-1$. Then,

$$F(i+n1, j+n2) = (i-1+n1).c + (j-1+n2).$$

Now, by the definition of N,

$$(i-1+n1).c + (j-1+n2) \leq (f-2+1).c + (c-2+1) = (f-1).c + c = f.c$$

Therefore, $F(i+n1, j+n2) \in D$. This argument can be repeated for every cell in the space.

Now, $\forall$ $d \in I_d$, $Z_{dj} \equiv Z \Leftrightarrow Z: Y_d \rightarrow X_j$. By the definition of the CC models presented previously, $Z: I_C \rightarrow I_D$. If the same mapping function is used, C is translated into $I_d$ (by definition of $I_d$). Let $F(E)$, with $E = (y_1,...,y_n)$ be the same mapping, defined as $F(E) = \sum_{j=1}^{n-1} \left[ \prod_{i=1}^{j} t_i \right] (y_{j+1} -1) + (y_1 -1)$. By applying the definition of the Z function,

$$F(E) = \sum_{j=1}^{n-1} \left[ \prod_{i=1}^{j} t_i \right] (y_{j+1} + v_{k\,j+1}) \bmod t_{j+1} + y_1 \bmod t_1 = \sum_{i=1}^{n} A_i .\, t_i, \text{ where } A_i \in [0, t_i-1] \Rightarrow F(E) \subseteq I_d.$$

For cell spaces in two dimensions, Z: $I_{ij} \rightarrow I_{pq}$. If the same mapping function is used, (i,j) is translated into $I_d$ (by definition of $I_d$). Let $F(p,q)$ be the same mapping, defined as $F(p,q) = (p-1).c + (q-1)$. By applying the definition of the Z function, $F(p,q) = [(p+r) \bmod f-1].c + [(q+s) \bmod c-1] = A.\,c + B$, where $A \in [0, f-1] \wedge B \in [0, c-1] \Rightarrow F(p,q) \subseteq I_d$.

Finally, using the same mapping function, it can be seen that select: $D \rightarrow D$.

Therefore, both models are equivalent, and the proposition is valid.

## 3.4 Formal specification of generic coupled Cell-DEVS models.

The procedure shown in the previous section allowed to define complete cell spaces using parameters. In this section, the coupling with other kind of models will be defined. The idea is to extend the definition of coupled models to allow the combination of Cell-DEVS and other basic models in a hierarchy. For instance, a Cell-DEVS model can be coupled with others with a different neighborhood or cell's behavior. It also could be combined with models that are not Cell-DEVS.

To allow this definition, generic Cell-DEVS coupled model can be represented as:

$$GCC = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z, select >$$

Here,

**Ylist** is the output coupling list;

**Xlist** is the output coupling list;

***I*** represents the definition of the interface for the modular model;

***X*** is the set of external input events;

***Y*** is the set of external output events;

**n** is the dimension of the cell space;

{$\mathbf{t_1,...,t_n}$} is the number of cells in each of the dimensions;

**N** is the neighborhood set;

**C** is the cell space;

**B** is the set of border cells;

**Z** is the translation function; and

**select** is the tie-breaking function for simultaneous events.

Here, for $\#T < \infty \wedge T \in \{N, Z, \boldsymbol{R}, \{0,1\}\} \cup \{\phi\}$;

- **Ylist** $\subseteq \{ (i_1,...,i_n) / i_k \in [1,t_k] \forall k \in [1,n], k \in \boldsymbol{N}\}$;
- **Xlist** $\subseteq \{ (i_1,...,i_n) / i_k \in [1,t_k] \forall k \in [1,n], k \in \boldsymbol{N}\}$;
- $\boldsymbol{I} = < P^X, P^y>$, where $\forall$ c $\in$ I, with **I** defined as in (5), $i \in \{X, Y\}$, $P_c{}^i$ is a definition of a port, and

  $P_c{}^i = \{ (N_c{}^i, T_c{}^i) \ / \ \forall c \in i\text{list}, \ N_c{}^i \in i(c)$ (port name), y $T_c{}^i \in T$ (port type)};
- $\boldsymbol{X} \subseteq T$;
- $\boldsymbol{Y} \subseteq T$;
- **n** $\in \boldsymbol{N}$, $n < \infty$;
- $\{\mathbf{t_1,...,t_n}\} \in \boldsymbol{N}$, with $t_k < \infty \ \forall k \in [1,n]$;
- For homogeneous neighborhoods, $\mathbf{N} = \{ (v_{k1},...,v_{kn}) / \forall (k \in N, k \in [1,\eta]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z}$

  $\wedge\ t_i - | v_{ki} | \geq 0 \}$. Here $\eta \in \boldsymbol{N}$, $\eta \leq \prod_{i=1}^{n} t_i \wedge \eta = \#N$. Instead, if general neighborhoods are defined,

  $\mathbf{N} = \{N_c / c \in \boldsymbol{N^n}, c \in [1,t1]x...x[1,t_n]\}$, with $N_c = \{ (v_{k1},...,v_{kn})_c / \forall (k \in N, k \in [1, \eta_c]) \wedge (i \in N, i \in [1, n]), v_{ki} \in \boldsymbol{Z} \ \wedge t_i - | v_{ki} | \geq 0 \}$. Here, $\eta_c \in N \ \wedge \eta_c = \#N_c$.
- $\mathbf{C} = \{ C_c / c \in \mathbf{I} \wedge C_c = < I_c, X_c, Y_c, S_c, N_c, d_c, \delta_{int_c}, \delta_{ext_c}, \tau_c, \lambda_c, D_c>$ is a TDC or IDC component, such as the defined previously, being **I** defined as in (5);
- $\mathbf{B} = \{\varnothing\}$ if the cell space is "wrapped", or

  $\mathbf{B} = \{C_b / C_b \in C$ with $b \in \mathbf{I}\}$, with **I** defined as in (4). Here, B is subject to the restriction that $\tau_b \neq \tau_c = \tau \ \forall (C_c \notin B) \wedge (C_b \in B)$. In most cases, $B = \{C_b / C_b \in C$ with $b \in L\}$, being $L = \{ (i_1,...,i_n) / i_j = 0 \vee i_j = t_j \ \forall j \in [1, n] \}$, and subject to the restriction that $\tau_b \neq \tau_c = \tau \ \forall c \notin L$.
- **Z**: $I_C \rightarrow I_D$ is the translation function, defined (using (4) ) as:

  $Z(P_C{}^{Y_q}) = P_D{}^{X_q}, \forall (q \in \boldsymbol{N}, q \in [1,\eta])$, where $\forall V_k \in N, D = C +_t V_k$; and

  $Z(P_C{}^{X_q}) = P_D{}^{Y_q}, \forall (q \in \boldsymbol{N}, q \in [1,\eta])$, where $\forall V_k \in N, D = C -_t V_k$.
- **select** $= \{ (s_1, ..., s_n) / s_i \in \boldsymbol{N}, \forall i \in [0, n] \}$.

The present specification allows to define automatically a modular Cell-DEVS space consisting of different submodels. The specification is independent of the simulation technique, and can be used to verify the model's correctness. Several differences can be found respecting the specification of closed Cell-DEVS models defined in the previous section. First, as in every coupled DEVS model admitting inputs and outputs, the sets ***X*** and ***Y*** have been included. As these models can be coupled with others, the ***I*** interface is also defined.

The **Z** function defined in the previous section is used to carry out the internal coupling of the cell space. Finally two new sets have been included (**Xlist** and **Ylist)** to define cells that receive or produce external events for the coupled model. These lists will be used to define automatically other input/output ports in the affected cells.

Finally, the **select** function is defined as a list of positions in the neighborhood. The list is ordered according the selection criteria to be used when more than one cell is active simultaneously.

**Lemma 3**

The GCC models are coupled DEVS models.

**Proof:**

It must be shown that the sets

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

and

$$GCC = < Xlist, Ylist, I, X, Y, N, n, \{t_1,...,t_n\}, C, B, Z, select >$$

are equivalent.

The equivalence of most of the sets was shown in the Lemma 2. Besides, $X' \equiv X$; and $Y' \equiv Y$, due to the definition of each of these sets. Finally, using the same mapping of the previous Lemma, it can be seen that Xlist $\subseteq$ D and Ylist $\subseteq$ D. Therefore both sets are equivalent, and the proposition is valid.

¦

## 3.5 Coupled DEVS models: redefinition to include Cell-DEVS models.

The original definition for DEVS coupled model has been changed so as to include base models that can be seen as cell spaces. For the Cell-DEVS models, the coupling scheme should use the previous input and output cell's definitions. Therefore, a coupled DEVS model will be defined as:

$$CM = < X, Y, D, \{M_d\}, \{I_d\}, \{Z_{dj}\}, select >$$

- ***X*** is the external input events set;
- ***Y*** is the external output events set
- **D** $\in$ ***N*** is an index for the components of the coupled model, and

$\forall$ d $\in$ D, $\mathbf{M_d}$ is a DEVS basic model, where

$$M_d = GCC_d = < I_d, X_d, Y_d, Xlist_d, Ylist_d, n_d, \{t_1,...,t_n\}_d, N_d, C_d, B_d, Z_d, select_d>$$

if the coupled model is Cell-DEVS, and

$$M_d = < X_d, S_d, Y_d, \delta_{int_d}, \delta_{ext_d}, D_d >$$

otherwise.

$\boldsymbol{I_d}$ is the set of models influenced by the model d, and $\forall\ j \in I_d$,

$\mathbf{Z_{dj}}$ is the translation function from d to j, where

$Z_{dj}$: $Y_d \rightarrow X_j$ if none of the implied models is Cell-DEVS, or

$Z_{dj}$: $Y(c_1)_d \rightarrow X(c_2)_j$, with $(c_1) \in$ Ylist$_d$, and $(c_2) \in$ Xlist$_j$ if any of the models d or j is a GCC.

Finally, **select** is the tie-breaking selector.

When a Cell-DEVS is executed, the $Z_{ij}$ function translates the outputs of a cell into inputs for other models using the two lists previously defined. The names of the input/output ports will be defined by using the contents of Xlist and Ylist.

**Lemma 4**

The CM models are coupled DEVS models.

**Proof:**

As it was shown, the GCC models are DEVS basic models, and all the sets here defined are therefore equivalent to those of coupled DEVS models. Therefore, it suffices with proving that the $Z_{dj}$ function is equivalent to that one defined for DEVS models. However, here $Y(c1)_d \subseteq Y_d$ and $X(c2)_j \subseteq X_j$ due to the definition of the lists Xlist and Ylist and the mapping function previously considered. Therefore, both sets are equivalent and the proposition is valid.

## 3.6 Closure under coupling of the Cell-DEVS models.

This section is devoted to show that the Cell-DEVS models are closed under coupling. This means that a Cell-DEVS coupled model can be associated with a DEVS basic model. This allows to integrate the model into the model's hierarchy, reusing submodels previ ously verified.

**Lemma 5**

The GCC models are closed under coupling.

**Proof:**

To proof the closure, is necessary to show that the coupled model

$$GCC = < I, X, Y, \text{Xlist}, \text{Ylist}, N, n, \{t_1,...,t_n\}, C, B, Z, \text{select} >$$

is equivalent to the coupled model

$$CM = < X, S, Y, \delta_{int}, \delta_{ext}, D >$$

Let $S = \underset{C\in [1,t1]\ x...x\ [1,\ tn]}{X} Q_C$

where $\forall\ C \in [1,t_1]x...x[1,t_n], Q_C = \{ (s_C, e_C) / s_C \in \theta_C \ x\ N_C\ x\ d_C; e_C \in [0, D_C(s_C)]\}$; where $C_C \in C$ is a TDC or IDC model.

D: $S \rightarrow \boldsymbol{R}_0^+ \cup \infty$, defined as $D(s) = \min \{\sigma_C / C \in [1,t_1]x...x[1,t_n]\ \forall\ C_C \in C, \sigma_C = D_C(s_C) - e_C\}$.

Using this definition of the duration function, the imminent components set can be defined as:

$IMM(s) = \{ C / c \in C \wedge \sigma_C = D(s) \}$

Let C* = select(IMM(s)). This is the component that will be executed.

$\delta_{int}$: $S \rightarrow S$. Let $s = (..., (s_C, e_C, N_C, d_C),...)$. Therefore, $\delta_{int}(s) = s' = (..., (s_C', e_C', N_C', d_C'),...)$, with

$$(s_C', e_C', N_C', d_C') = \begin{cases} (\delta_{int\ C}(s_C), 0, N_C, d_C) & \text{if } C = C^* \\ (\delta_{ext\ D}( (s_D, e_D + D(s) ), X_D), \ 0, N_D, d_D) & \text{if } D \in N_{C^*} \\ (s_C, e_C + D(s), N_C, d_C) & \text{otherwise.} \end{cases}$$

where $X_D = Z_{C^*, D} (\lambda_{C^*}(s_{C^*}) )$

This means that the internal transition function changes the status of the imminent cell D* according with its internal transition function. Then, it updates all the influenced components according with the inputs produced by the output of D*. In the other components only the elapsed time is updated.

$\delta_{ext}$: $QxX \rightarrow S$

$\delta_{ext}( (s, e), x) = s' = (...,(s_C', e_C', N_C', d_C'), ...)$ with

$$(s_C', e_C', N_C', d_C') = \begin{cases} (\delta_{ext\ D} (s_D, e_D + e, Z_{C,\ D}(x)), 0, N_C, d_C) & \text{if } D \in N_D \\ (s_C, e_C + e, N_C, d_C) & \text{otherwise.} \end{cases}$$

In this case all the cells influenced by the external input x change their status according with the input translated using the mapping between interfaces. The remaining cells only modify their elapsed time.

Finally, $\lambda$: $S \rightarrow Y$.

$$\lambda(s) = \begin{cases} Z_{C^*, D}(\lambda_{C^*}(s_{C^*})) & \text{if } C^* \in I_D \\ \varnothing & \text{otherwise.} \end{cases}$$

That is, if (i,j)* sends an output, it is translated into an input to the coupled model using the mapping between interfaces $Z_{C^*, D}$.

## 4. CONCLUSION

In this work the definition for n-dimensional Cell-DEVS models was presented. The formalism allows to define complex model using a formal approach, allowing to verify automatically the correctness of the models. The relationship with other DEVS models was shown, allowing the integration between cell spaces and other DEVS models. As it was also shown that the models are closed under coupling, they can be integrated in a DEVS modelling hierarchy. One of the main contributions is related with the definition of complex timing behavior for the cells in the space using very simple constructions. Transport and inertial delays allow the modeler to make easier the timing representation of each cell in the space.

The model definition is independent of the simulation mechanism, and at present several simulation techniques have been implemented. In (Wainer and Giambiasi 2001 b.), the simulation mechanism defined in (Zeigler 1990) was extended to include cell spaces. A second mechanism, defining flat cell spaces was also implemented, allowing seven-fold improvements in the execution times. Recently, a theory of DEVS quantized models was developed (Zeigler et al., 2000). The theory has been verified when applied to predictive quantization of arbitrary ordinary differential equation models. Quantized models reduce substantially the frequency of message updates. Quantized Cell-DEVS have been implemented, allowing to check the theoretical results of quantified DEVS models when applied to Cell-DEVS (Wainer and Zeigler, 2000).

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. The formal specification has shown improvements in the development times of up to ten-fold. The main improvements have been detected in the testing and maintenance phases, because the formal specification mechanism allowed to reduce the resources devoted to these phases.

The specifications presented in this work were used as the formal basis to implement a modelling and simulation tool allowing the user to define timed Cell-DEVS models in uniprocessors (Rodríguez and Wainer 1999). This tool includes an specification language that lets the user to define simulation models using the formal specifications showed in sections 2 and 3.

The simulation literature showed that the use of parallel simulation mechanisms is a promising approach to obtain results, because it allows speedups in the simulation process. This is the case for Cell-DEVS models, because they involve a high degree of computation time. The original definition has been recently been reformulated to accept Cell-DEVS that can be executed in parallel. As stated in (Chow and Zeigler, 1994), if we call **e** to the elapsed time since the occurrence of an event, a model can exist at e=0 or e=D(s). In coupled models, the modeler can use the select function to resolve the conflicts of simultaneous events. The case is different for basic models: once they are coupled, ambiguity arises when a model scheduled for an internal transition receives an event. The problem here is how to determine which of both elapsed times should be used. The extension permits parallel specification of these models, including a confluent local computing function that permits a user to specify the course of action under simultaneous events (Wainer 2000). At present the simulation mechanism has been defined to permit parallel model execution, and has been implemented successfully (Troccoli and Wainer 2001).

## REFERENCES

CHOW, A.; ZEIGLER, B. "Abstract Simulator for the parallel DEVS formalism". *Proceedings of the Winter Simulation Conference*. 1994.

GHOSH, S.; GIAMBIASI, N. "On the need for consistency between the VHDL language constructions and the underlying hardware design". Proceedings of the 8th. European Simulation Symposium. Genoa, Italy. Vol. I. pp. 562-567. 1996.

GIAMBIASI, N.; MIARA, A. "SILOG: A practical tool for digital logic circuit simulation. Proceedings of the 16th. D.A.C., San Diego, U.S.A. 1976.

MOON, Y.; ZEIGLER, B.; BALL, G.; GUERTIN, D. P. "DEVS representation of spatially distributed systems: validity, complexity reduction". *IEEE Transactions on Systems, Man and Cybernetics*. pp. 288-296. 1996.

RODRIGUEZ, D.; WAINER, G. "New Extensions to the CD++ tool". In *Proceedings of SCS Summer Multiconference on Computer Simulation*. Chicago, U.S.A. 1999

TOFFOLI, T.; MARGOLUS, N. "Cellular Automata Machines". *The MIT Press*, Cambridge, MA. 1987.

TROCCOLI, A.; WAINER, G. "Performance analysis of cellular models with Parallel Cell-DEVS". Accepted for publication in *Proceedings of 35th Summer Computer Simulation Conference*, Orlando, Florida, USA. 2001.

WAINER, G. “Improved cellular models with parallel Cell-DEVS”. In *Transactions of the SCS.* June 2000.

WAINER, G.; GIAMBIASI, N. "Timed Cell-DEVS: modelling and simulation of cell spaces ". In "Discrete Event Modeling & Simulation: Enabling Future Technologies", to be published by Springer-Verlag. 2001.

WAINER, G.; GIAMBIASI, N. b. "Application of the Cell-DEVS paradigm for cell spaces modelling and simulation.". *Simulation*. January 2001.

WAINER, G.; ZEIGLER, B. "Experimental results of Timed Cell-DEVS quantization". In *Proceedings of AIS'2000*. Tucson, Arizona. U.S.A. 2000.

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

ZEIGLER, B. "Theory of modeling and simulation". Wiley, 1976.

ZEIGLER, B. "Multifaceted Modeling and discrete event simulation". Academic Press, 1984.

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

ZEIGLER, B., CHO, H.; LEE, J.; SARJOUGHIAN, H. "The DEVS/HLA Distributed Simulation Environment And Its Support for Predictive Filtering". DARPA Contract N6133997K-0007: ECE Dept., UA, Tucson, AZ. 1998.

ZEIGLER, B.; KIM, T.; PRAEHOFER, H. "Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems". Academic Press. 2000.

***Gabriel A. Wainer*** received the M.Sc. (1993) and the Ph.D. degrees (1998, with highest honours) at the Universidad de Buenos Aires, Argentina, and DIAM/IUSPIM, Université d'Aix-Marseille III, France. He is Assistant Professor at the Systems and Computer Engineering, Carleton University (Ottawa, Canada). He was Assistant Professor at the Computer Sciences Dept. of the Universidad de Buenos Aires, Argentina, and a visiting research scholar at the Arizona Center of Integrated Modelling and Simulation (ACIMS, University of Arizona). He has published more than 50 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 has been the PI of several research projects, and participated in different international research programs. Prof. Wainer is a member of the Board of Directors of The Society for Computer Simulation International (SCS). He is the coordinator of a group on DEVS standardization. He has been appointed Associate Editor of the Transactions of the SCS. He is also a Co-associate Director of the Ottawa Center of The McLeod Institute of Simulation Sciences.

***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 PhD 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.