# Cellular Models for Emerging Traffic Behavior

**Gabriel Wainer[1], Cristina Ruiz-Martin[1,2], Adolfo Lopez-Paredes[2]**

[1] Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada
[2] INSISOC. Universidad de Valladolid. Spain

**Abstract** Various researches used cellular models in which we can observe emerging behaviors that have not been programmed explicitly. The Cell-DEVS methodology is formal modeling technique that permits defining each cell in a cell space as individual independent entity. We used Cell-DEVS to build different models of traffic. We show how to model cell spaces with emerging behavior using this methodology. We present basic models for roundabouts, traffic monitoring, railways, and traffic shockwaves based on cellular models.

*Keywords: Cell-DEVS, Cellular Automata, Traffic Modeling*

## Introduction

The application of modeling and simulation techniques to study both natural and human–made systems shows an increasing trend due to its advantages. Traditional approaches to solve these problems used differential equations that were solved analytically. However, these traditional approaches have some restrictions when modeling complex artificial systems. The introduction of new methods based on Cellular Automata (CA) has provided new ways to study complex systems that can be represented as cell spaces [1].

A CA is an infinite regular n-dimensional grid whose cells can take one value form a finite set (cell state). The cell states are updated in discrete time steps following local rules in a simultaneous and synchronous way. These local rules are based on the actual cell state and a finite set of neighboring cells. Cell-DEVS [2] is a formalism that combines CA with Discrete Events Systems Specifications (DEVS). DEVS [3] is a formalism for modeling discrete-event dynamic systems that allows for hierarchical decomposition of the model by defining a way to couple existing DEVS models. In Cell-DEVS, as in CA, we define a grid of cells that has a state and a set of local rules to calculate the next state based on the actual cell state and a finite set of neighboring cells. The difference is that the behavior of each cell is defined as a DEVS atomic model and the CA as a coupled model. Cell-DEVS is more flexible than CA as it can be easily coupled with other DEVS models, and we can define timing delays for each cell, making the timing specification more expressive.

Both DEVS and Cell-DEVS formalisms can be implemented in the CD++ environment [2]. This environment has been used successfully to develop different types of systems: biological (ecological models, heart issue, ant foraging systems, fire spread, etc.), physical (diffusion, binary solidification, excitable media, surface tension, etc.), artificial (robot trajectories, networking,

traffic, etc.), and others [2, 4-6]. The techniques we used in CD++ enable the models execution using simulation engines that are completely independent from the modeling aspects.

In this paper, we show how to use this methodology to study traffic behavior. For this purpose, we build different traffic models and show emerging behaviors that have not been explicitly programmed. In the following sections, we will introduce how to use the Cell-DEVS formalism, and we will show how to model complex cell spaces with emerging behavior using this methodology. We will present basic models for roundabouts, traffic monitoring, railways, and traffic shockwaves, based on cellular models.

## Background

The DEVS formalism [3] provides a framework to develop hierarchical models in a modular way, allowing model reuse and thus, reducing development time and testing. A DEVS model is defined as a black box with a state and a duration for that state. When state duration time elapses, an output event is sent, and an internal transition takes place to change the model state. A state can also change when an external event is received. These single models are called atomic. DEVS atomic models can be put together by linking the outputs of a model to inputs of other models to form coupled models.

Based on the DEVS formalism, Cell-DEVS defines a cell space as DEVS atomic models. A Cell-DEVS atomic model is defined as follow [2]:

$$\text{Cell} - \text{DEVS AM} = < \text{X}, \text{Y}, \text{I}, \text{S}, \theta, \text{E}, \text{delay}, \text{d}, \delta_{\text{ext}}, \delta_{\text{int}}, \tau, \lambda, \text{D} >$$

Where:

$X$: Is the set of input events.

$Y$: Is the set of output events.

I: Represents the model's modular interface.

S: Is the set of sequential states for the cell.

$\theta$: Is the set of the cell's state variables.

E: Is the set of states for the input events to compute the future state.

*delay*: is the type of delay in the cell to send its value to the neighbors. The delay can be transport or inertial. In transport delays, the future value will be added to a queue sorted by output time. In inertial delays, any previous scheduled output values will be preempted and only the new one will be scheduled.

d: Is the delay for the cell.

$\delta_{int}$: Is the internal transition function.

$\delta_{ext}$: Is the external transition function. This function actives the local computation function ($\tau$)

$\tau$: Is the local computation function (i.e. the rules to calculate the next state)

$\lambda$: Is the output function.

D: Is the state's duration function.

Once the behavior of each cell is defined, the cell space is built as a Cell-DEVS coupled model:

$$\text{Cell} - \text{DEVS CM} = < \text{Xlist}, \text{Ylist}, \text{I}, \text{X}, \text{Y}, \text{n}, \{\text{t}_1, \dots, \text{t}_\text{n}\}, \text{N}, \text{C}, \text{B}, \text{Z}, \text{select} >$$

Where:

Xlist: Is the input coupling list.

Ylist: 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.

*{$t_1$,...,$t_n$}:* Is the number of cells in each of the dimensions.

N: Is the neighborhood set. It defines the connections between cells.

C: Is the cell space. The cell space is finite.

B: Is the set of border cells. The border cells may have a different neighborhood or the space can be "wrapped" (i.e. the cells in one border are connected with the ones in the other)

Z: Is the translation function to define the external and internal coupling of cells.

select: Is the tie-breaking function for simultaneous events.

The CD++ tool allows implementing Cell-DEVS models with applications in numerous fields [2]. Here we explain how to implement a simple CA. Based on this example, in the next sections, we explain the application for modeling traffic.

The Brian's Brain CA [7] is an extension of the Seeds pattern model. The model's cells can be in any of three states: *firing or on*, *refractory or dying*, and *dead or off*. The model uses Moore's neighborhood (i.e. the neighbor cells are the eight adjacent cells)

The CA follows three simple rules. At each time step, (1) a *dead* cell turns to *firing* if it has exactly two firing neighbors; (2) a *firing* cell always evolves to *refractory*; and (3) a *refractory* cell always evolves to *dead*. Following these simple rules, the CA shows an emergent behavior. The refractory cells tend to lead to a pattern that reappears after a certain amount of generations in the same orientation but in a different position. However, many Brian's Brain patterns will explode messily and chaotically, but most of them will often contain diagonal waves of firing and refractory cells. These patterns cannot be predicted in advance.

This simple CA can be specified as a Cell-DEVS model as follow:

$$Brian's\ Brain\ AM = < X, Y, \text{I}, S, \theta, \text{E}, \text{delay}, \text{d}, \delta_{ext}, \delta_{int}, \tau, \lambda, D >$$

Where:

X=Y={0,1,2}

S={s|s∈{0,1,2}} // 0 = dead or off; 1 = firing or on; 2 = refractory or dying where 0 means dead or off, 1 means firing or on and 2 means refractory or dying.

I=Represents the model's modular interface, defined by Cell-DEVS as follows:

$I = < \eta, \mu^x, \mu^y, P^x, P^y >$, where $\eta$=9 is the neighborhood's size, $\mu^x = \mu^y = 1$ is the number of input/output ports. Therefore, $P^x = \{P_1^x, P_2^x, P_3^x, P_4^x, P_5^x, P_6^x, P_7^x, P_8^x, P_9^x\} =$

$\{(N_1^x, integer), (N_2^x, integer)(N_3^x, integer)(N_4^x, integer)(N_5^x, integer),$

$(N_6^x, integer)$, $(N_7^x, integer), (N_8^x, integer)$, $(N_9^x, integer)$}and
$P^y = \{P_1^y, P_2^y, P_3^y, P_4^y, P_5^y, P_6^y, P_7^y, P_8^y, P_9^y\} =$
$\{(N_1^y, integer), (N_2^y, integer), (N_3^y, integer), (N_4^y, integer),$
$(N_5^y, integer), (N_6^y, integer), (N_7^y, integer), (N_8^y, integer),$
$(N_9^y, integer)\}$.

$\theta = \{(s, phase, \sigma queue, \sigma) |$ sϵS is the status value for the cell, phase ϵ {active, passive}, $\sigma queue = \{((s_1, \sigma_1), \dots, (s_m, \sigma_m)) | (m \epsilon N \wedge m < \infty) \wedge \forall (i \epsilon N, i \epsilon [1, m]), s_i \epsilon S \wedge \sigma_i \epsilon R_0^+ \cup \infty\}$ and $\sigma \epsilon R_0^+ \cup \infty\}$

E = {(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)}

*delay*: transport.

d: 100ms

τ: is defined by the following rules:

(0,0)=0 **if** (0,0)=2

(0,0)=1 **if** ((0,0)=0) & ((number of neighbors with s=1)=2)

(0,0)=2 **if** (0,0)=1

δint, δext, λ and D are defined using Cell-DEVS specifications.

Once the behavior of each cell is defined, the cell space is built as a Cell-DEVS coupled model:

$Brian'sBrain\ CM = < Xlist, Ylist,$ I, X, Y, n, $\{t_1, \dots, t_n\}$, N, C, B, Z, select >

Where:

X=Y=Xlist=Ylist=∅

I=∅ as this is a closed model;

n=2

$\{t_1,\dots,t_n\}=\{20,20\}$

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

C = *Brian's Brain AM*

B=∅ //wrapped

Z={

$P_{i,j}^{Y1} \rightarrow P_{i,j-1}^{X1}$ $P_{i,j+1}^{Y1} \rightarrow P_{i,j}^{X1}$

$P_{i,j}^{Y2} \rightarrow P_{i+1,j}^{X2}$ $P_{i-1,j}^{Y2} \rightarrow P_{i,j}^{X2}$

$P_{i,j}^{Y3} \rightarrow P_{i,j+1}^{X3}$ $P_{i,j-1}^{Y3} \rightarrow P_{i,j}^{X3}$

$P_{i,j}^{Y4} \rightarrow P_{i-1,j}^{X4}$ $P_{i+1,j}^{Y4} \rightarrow P_{i,j}^{X4}$

$P_{i,j}^{Y5} \rightarrow P_{i,j}^{X5}$ $P_{i,j}^{Y5} \rightarrow P_{i,j}^{X5}$

$P_{i,j}^{Y6} \rightarrow P_{i-1,j-1}^{X6}$ $P_{i-1,j-1}^{Y6} \rightarrow P_{i,j}^{X6}$

$P_{i,j}^{Y7} \rightarrow P_{i-1,j+1}^{X7}$ $P_{i-1,j+1}^{Y7} \rightarrow P_{i,j}^{X7}$

$P_{i,j}^{Y8} \rightarrow P_{i+1,j-1}^{X8}$ $P_{i+1,j-1}^{Y8} \rightarrow P_{i,j}^{X8}$

$P_{i,j}^{Y9} \rightarrow P_{i+1,j+1}^{X9}$ $P_{i+1,j+1}^{Y9} \rightarrow P_{i,j}^{X9}$ }

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

To implement the model in CD++, we need to map the information from the formal model definition to define the possible cells states, the neighborhood set, the rules, the size of the cell space, the type and value of the delay, the type of borders and the initial value of the cells. All these parameters are defined as shown as in the code below:

```
[briansbrain]
type: cell              width: 20         height: 20
delay: transport        border: wrapped
neighbors: (-1,-1),(-1,-0),(-1,1)
neighbors: (0,-1),(0,-0),(0,1)
neighbors: (1,-1),(1,-0),(1,1)
localtransition: briansbrain-rule

[briansbrain-rule]
rule : 0 100 { (1,0)=2 }   (0,0) = 2
rule : 1 100 { (0,0)=1 and trueCount = 2 }
rule : 2 100 { (0,0)=1 }
rule: 0 100 {t}
```

**Fig. 1.** Brian's Brain CD++ .ma file for a 20x20 grid

In figure 1, we show the implementation in CD++ of Brain's Brain model in a 20x20 grid. First, we define the name of the Cell-DEVS component. Then, we define the type of model (a cell space) and its size (as it is a two dimensional model, we only need to define its width and height). The model uses a transport delay, and wrapped borders. After the keyword "neighbors", we define the neighborhood set (in this case, a Moore's neighborhood). Finally, the local transition function is defined as a set of rules. They are implemented following CD++ high-level language with the form:

rule: POSTCONDITION DELAY { PRECONDITION }

These indicate that when the PRECONDITION is satisfied, the state of the cell will change to the designated POSTCONDITION, whose computed value will be transmitted to other components after consuming the DELAY. If the precondition is false, the next rule in the list is evaluated until a rule is satisfied or there are no more rules.

In figure 2, we can see a snapshot of the Brian's Brain CA simulation in CD++ with a random initial configuration in the center of a 400x400 grid. Dead cells are represented in black, firing cells are represented in white and refractory or dying cells in blue (light gray on the paper). In the figure, we can easily appreciate the diagonal waves of firing and refractory cells that appear as emergent behavior. Despite in the initial configuration, firing and refractory cells only were defined in the center of the grid, they expand and move around the whole grid after few steps. The full simulation is available in https://www.youtube.com/watch?v=8MCDuRVEbeg. In this video, we can see how the diagonal waves that appear as emergent behavior move and change over time. After 6000 steps, we cannot see a pattern that repeats over time. In smaller grids, this pattern appears.

In the next sections, we will show how to apply this methodology and CD++ to study the emergent behavior of different traffic rules and problems.

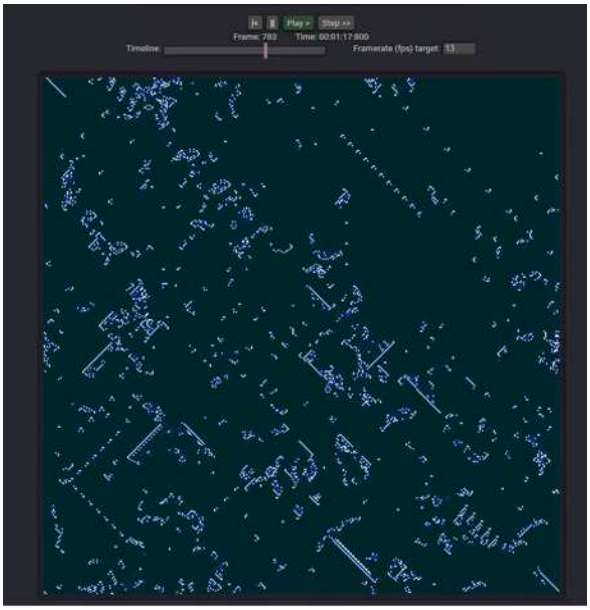

**Fig. 2.** Snapshot of Brian's Brian rules CA implemented in CD++

## Simple Cell-DEVS models for Traffic

Traffic problems are increasing every day, and in order to control and manage the increasing amount of vehicles and avoid major problems, policy makers set up different policies. As stated in the introduction, the introduction of new methods based on CA has provided new ways to study this type of complex systems that can be represented as a cell space [1]. In this section, we show how we can implement very simple traffic models in Cell-DEVS in CD++. We also show how we can nicely visualize the simulation results. This visualization allows us to study the emergent behavior of the implemented rules. Moreover, it allows us to provide understandable results to police makers without programming background.

### *Traffic Flow Using Rule-184 and Track Combination*

The rule-184 for one-dimensional Cellular Automata has been proposed as a simple prototype for describing traffic flow. The earliest research on Rule 184 was presented in [8,9]. The idea is to consider vehicles as particles and roads as discrete grids. Particles can move forward at the same time in each time step as long as the next sites in the direction of advance are empty. This model can be used to illustrate the phase transition between a free movement phase and a jammed phase.

We can formally specify this model a Cell-DEVS as follow:

$$Rule-184\ AM = < X, Y, \mathrm{I}, S, \theta, \mathrm{E}, \mathrm{delay}, \mathrm{d}, \delta_{ext}, \delta_{int}, \tau, \lambda, D >$$

X=Y={0,1}
S={s|s∈{0,1}} // 0 = empty and 1 = occupied by a vehicle.
I and θ are defined as above in Brain's Brain model
E={(0,-1),(0,0),(0,1)}
*delay*: transport.
d: 100ms
τ: is defined by the following rules:

(0,0)=1 **if** {(0,-1)=1 and (0,0)=1 and (0,1)=1}
(0,0)=0 **if** {(0,-1)=1 and (0,0)=1 and (0,1)=0}
(0,0)=1 **if** {(0,-1)=1 and (0,0)=0 and (0,1)=1}
(0,0)=1 **if** {(0,-1)=1 and (0,0)=0 and (0,1)=0}
(0,0)=1 **if** {(0,-1)=0 and (0,0)=1 and (0,1)=1}
(0,0)=0 **if** {(0,-1)=0 and (0,0)=1 and (0,1)=0}
(0,0)=0 **if** {(0,-1)=0 and (0,0)=0 and (0,1)=1}
(0,0)=0 **if** {(0,-1)=0 and (0,0)=0 and (0,1)=0}

δint, δext , λ and D are defined using Cell-DEVS specifications.

The Cell-DEVS coupled model is defined as:

$Rule-184\ CM = < Xlist, Ylist, \mathrm{I, X, Y, n}, \{t_1, \dots, t_n\}, \mathrm{N, C, B, Z, select} >$

X = Y = Xlist = Ylist = ∅
I is defined as in Brain's Brain model
n= 2; $\{t_1,\dots,t_n\} = \{1,20\}$
N={ (0,-1),(0,0),(0,1)}
C= is an array of $Rule-184\ \ AM$
B= ∅ // non-wrapped
Z is defined as above based on N;
select=N={(0,-1),(0,0),(0,1)}

Implementing the Rule-184 in CD++, we can simulate different initial values configurations and study how this configurations influence the traffic. In Figure 3, we show different steps during the simulation. Red cells represent that it is occupied by a vehicle. Gray cells represent that the cell is empty. Through the different simulation steps, we can see how vehicles move forward. However, they are blocked when there are vehicles in front of them.

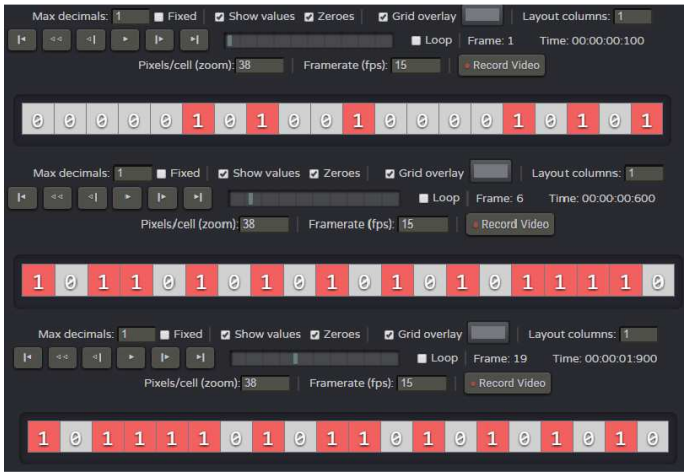


**Fig. 3.** Traffic simulation results: Rule-184. Random initial values at three different time steps.

The Rule-184 can be extended to study traffic with two lanes. Adjusting the above model, the neighbors for the first line are as show in Figure 4.a (the front cell, the back cell, the cell below and itself). The second line has five neighbors as shown in Figure 4.b.

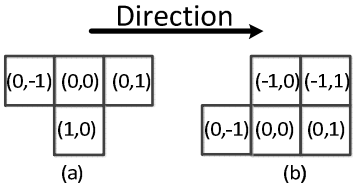


**Fig. 4.** Neighborhood of the first row and second row and the direction of the movement.

By combining the rule-184 methods and the vehicle moving procedure, the new rules are as follows:

```
(0,0)=0 {(1,0)=0 and (0,0)=1 and cellpos(0)=0} %move down
(0,0)=0  {(1,0)=1  and  (0,-1)=0  and  (0,0)=0  and  (0,1)=0  and
   cellpos(0)=0} %no vehicle
(0,0)=0  {(1,0)=1  and  (0,-1)=0  and  (0,0)=0  and  (0,1)=1  and
   cellpos(0)=0} %vehicle ahead
(0,0)=0  {(1,0)=1  and  (0,-1)=0  and  (0,0)=1  and  (0,1)=0  and
   cellpos(0)=0} %move ahead
(0,0)=1  {(1,0)=1  and  (0,-1)=0  and  (0,0)=1  and  (0,1)=1  and
   cellpos(0)=0} %stuck
(0,0)=1  {(1,0)=1  and  (0,-1)=1  and  (1,-1)=1  and  (0,0)=0  and
   cellpos(0)=0} %vehicle behind
        …
```

After the delay, the high priority vehicle from the first row can move to the second row if there is space. In turn, if the vehicle in the second row detects a vehicle in front of it, the vehicle would not move and will stay in its position. The rest of rules describe the inner rule of each row, which is the implementation of the rule-184.

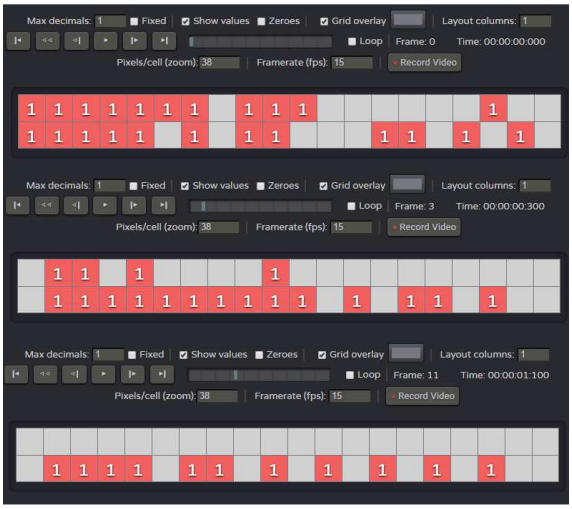


**Fig. 5.** Traffic simulation results with Rule-184 extended for track combination and random initial values at three different time steps

Figure 5 shows some simulation results. Red cells represent spaces occupied by vehicles. Through the different simulation steps, we can see how vehicles move forward and change lanes when needed. These models show how to define precise rules of priority. Different behaviors for vehicle movement can be implemented, compared and analyzed to make more appropriate traffic policies.

## *Shockwaves on Highways*

This section presents a model for shockwaves on highways, using a homogenous highway at the initial state. We allow traffic density to change slowly. The traffic initially moves at free flow speed, and then we add a disturbance to the flow. To study how the traffic flow evolves in a two-lane one-way road segment taking into account traffic density, we take make the following modeling considerations:

- The incoming vehicle time follows a Poisson distribution and the rate of incoming vehicles increases with time.
- Boundary conditions are specified by means of input and output cells. An output cell is a sink for exiting traffic. Source cells provide vehicles that discharge into an empty cell.
- The model is non-wrapped. Therefore, the vehicles in the upper and lower lane can change from one lane to the other avoiding crashes.

The Cell-DEVS specification of the model is as follow:

$$Shockwave\ AM = < X, Y, \mathrm{I}, S, \theta, \mathrm{E}, \mathrm{delay}, \mathrm{d}, \delta_{ext}, \delta_{int}, \tau, \lambda, D >$$

X = Y = {0,1,2}

S={s|s∈{0,1,2}} // 1 represents vehicles in the left lane; 2 represents vehicles in the right lane; 0 means no vehicle

I and θ are defined as above

E={(-1,-1),(-1,0),(-1, 1),(0, -1),(0, 0),(0, 1),(1, -1),(1, 0),(1, 1)}

*delay*: transport.

d: 1ms

τ: is defined by the following rules:

(0,0)=0 **if** (0,0)=1 and (-1,0)=0
(0,0)=1 **if** (0,0)=0 and (1,0)=1
(0,0)=0 **if** (0,0)=2 and (-1,0)=0
(0,0)=2 **if** (0,0)=0 and (1,0)=2
(0,0)=0 **if** (0,0)=1 and (-1,0)=1 and (0,1)=0 and (-1,1)=0
(0,0)=2 **if** (0,0)=0 and (1,0)=0 and (1,-1)=1 and (0,-1)=1
(0,0)=0 **if** (0,0)=2 and (-1,0)=2 and (0,-1)=0 and (-1,-1)=0
(0,0)=1 **if** (0,0)=0 and (1,0)=0 and (1,1)=2 and (0,1)=2

The Cell-DEVS coupled model is:

$$Shockwave\ CM = < Xlist, Ylist, \mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{n}, \{t_1, \ldots, t_n\}, \mathrm{N}, \mathrm{C}, \mathrm{B}, \mathrm{Z}, \mathrm{select} >$$

X = Y = {0,1}

Xlist={(29,0),(29,1)}

Ylist=∅

I is defined as above

n=2.

$\{t_1,\ldots,t_n\} = \{30,2\}$ // These parameters of the model change as we change the size of the cell space.

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

C =$Shockwave\ \ AM$

B = ∅ // non-wrapped

Z = is defined as above

select=N

Finally, let us consider the specification for the complete model, as we have two vehicle generators (incomingVehicles1, incomingVehicles2).

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

X= Y= ∅

D= {shockwave CM, incomingVehicles1, incomingVehicles2}, and ∀ i ∈ D, Mi is one of the basic DEVS models previously defined

I i is the set of influences of model i. In this case,

$I_{\text{shockwave CM}}$ = {incomingVehicles1, incomingVehicles2}

$I_{\text{incomingVehicles1}}$=$I_{\text{incomingVehicles2}}$=∅

$Z_{ij}$ = {

$Z_{\text{incomingVehicles1,shockwave CM}} : OUT \rightarrow X(0,29)_{\text{shockwave CM}}$

$Z_{\text{incomingVehicles2,shockwave CM}} : OUT \rightarrow X(1,29)_{\text{shockwave CM}}$ }

select = {incomingVehicles1, incomingVehicles2, shockwave CM }

There are two inputs, one in each line, representing the random arrival of vehicles. The interval time of vehicles is exponentially distributed. However, in order to examine the different rate of incoming vehicle, the mean in the vehicle generator can be changed. Each lane also has one sink, representing how the vehicles leave the highway. Different scenarios for traffic density arrival and different initial traffic conditions in the road can be simulated.

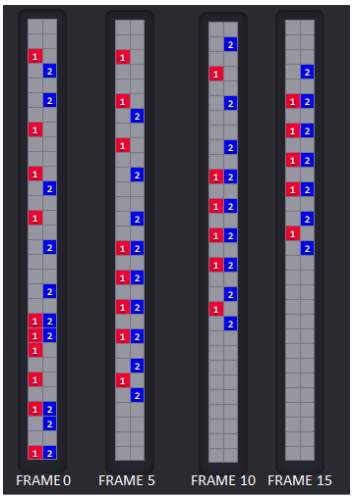


**Fig. 6.** Traffic simulation results for shockwaves with random initial values

Figure 6 shows a simulation scenario with a random initial distribution. Red cells represent that it occupied by a vehicle in the first line. Blue cells represent that the cell is occupied by a vehicle in the second row. Gray cells represent that

the cell is empty. At the beginning, the traffic density on both road segments is high and at the end, it becomes free-flow.

After showing some basic models with simple emergent behavior, the following sections will present more advanced traffic models: an example focusing on traffic in roundabouts and for traffic monitoring in highways.

## Modeling Roundabouts

This section shows a model of a roundabout with eight single lanes connected to a single-lane ring. In the ring, vehicles move counter-clockwise. The roundabout can be modeled as a Cell space as shown in Figure 7. The arrows represent the direction of the vehicle movement in the cell space. Blue cells represent the point where a vehicle can decide continuing going on in the roundabout or take the exit.

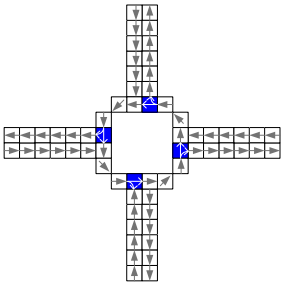

**Fig. 7.** The Model and the Direction Flow for Each Cell

To specify the model, the cell space is divided into 25 zones as shown in figure 8. As a cell in each zone behaves differently, the traffic rules are defined by zones. In the figure, white cells represent the traffic lanes and green cells represent the area near the roundabout where no vehicles are allowed. Four cells contain vehicle generators (one in each incoming line to the roundabout) to represent the incoming vehicles.

The Cell-DEVS model is as follow:

$8LineRoundabout\ AM = < X, Y, \mathrm{I}, S, \theta, \mathrm{E}, \mathrm{delay}, \mathrm{d}, \delta_{ext}, \delta_{int}, \tau, \lambda, D >$

X = Y = {0, 1, 2}

S = {s | s ∈ {0, 1, 2}} // 0 = empty, 1 = vehicle, 2 = no vehicles allowed

I and θ are defined as above

E={(-2,0),(-2,1),(-1,-2),(-1,-1),(-1,0),(-1,1),(0,-2),(0,-1),(0,0),(0,1),(0, 2),(1,-1), (1,0), (1,1),(1,2),(2,-1),(2,0)}

*delay*: transport.

d: 100ms

τ: is defined by the following rules:

*[roundabout-rule]*

(0,0) = 2 **if** {(0,0)=2}

*[R-lane-to-roundabout-rule]*

(0,0) = 0 **if** {(0,0)=1 and (0,-1)=0 and (-1,-1)=2}
(0,0) = 0 **if** {(0,0)=1 and (0,-1)=0 and (-1,-1)!=2 and (1,-1)=0 and (2,-1)=0}
(0,0) = 1 **if** {(0,0)=0 and (0,1)=1}
(0,0) = 1 **if** {(0,0)=1 and (0,-1)=1}
(0,0) = 1 **if** { (0,0)=1 and (1,-1)=1}
(0,0) = 1 **if** { (0,0)=1 and (2,-1)=1}
*[R-lane-from-roundabout-rule]*
(0,0) = 0 **if** {(0,0)=1 and (0,1)=0}
(0,0) = 1 **if** {(0,0)=0 and (0,-1)=1 and (1,-1)=2}
(0,0) = 1 **if** {(0,0)=0 and (0,-1)=1 and (1,-1)!=2 and ( random > 0.5)}

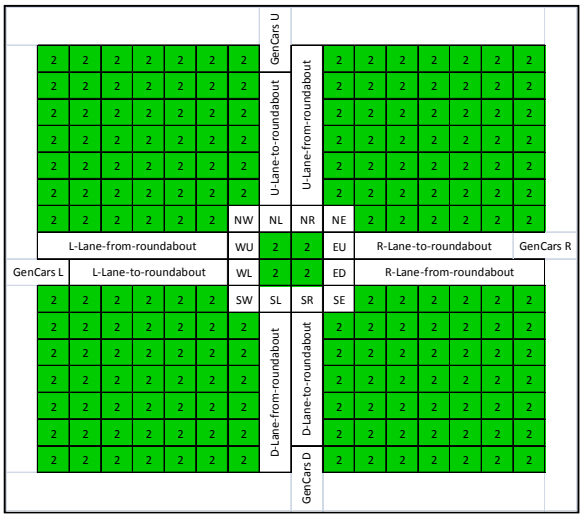


**Fig. 8.** Names of the zones in Traffic roundabout model.

The Cell-DEVS coupled model is:

$8LineRoundabout\ CM = < Xlist, Ylist,$ I, X, Y, n, $\{t_1, \dots, t_n\}$, N, C, B, Z, select $>$

X = Y = Xlist = Ylist = ∅
I is defined as above
n = 2
$\{t_1,\dots,t_n\} = \{16,16\}$
N={(-2,0),(-2,1),(-1,-2),(-1,-1),(-1,0),(-1,1),(0,-2),(0,-1),(0,0),(0,1), (0, 2),(1,-1),(1,0), (1,1),(1,2),(2,-1),(2,0)}
C = $8LineRoundabout\ \ AM$
Z is defined as above
select=N

Different initial traffic configurations can be simulated. Here we show two different cases: one when there are no yield signs and one with yield signs that give vehicles inside a roundabout the right to move and control traffic coming toward the ring.

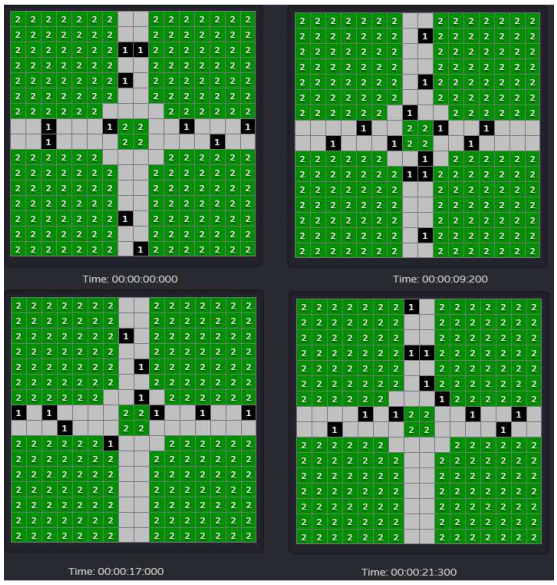


**Fig. 9.** Traffic simulation results in roundabout without yield signs

In figure 9, we show the simulation results without yield signs and in figure 10 with yield signs. In both figures, the green cells do not change during the simulation. Gray cells represent empty cells and black cells represent occupied cells by a vehicle. We can see various accidents where vehicles come to the roundabout and vehicles inside the roundabout meet (see for example time 9.200, where there is a vehicle coming and a vehicle in the roundabout close to it).

When we introduce yield sings, we obtain the results shown in figure 10. We can see that inside the roundabout all vehicles keep the safety distance, represented by an empty cell (gray) between vehicles. The accidents disappear.

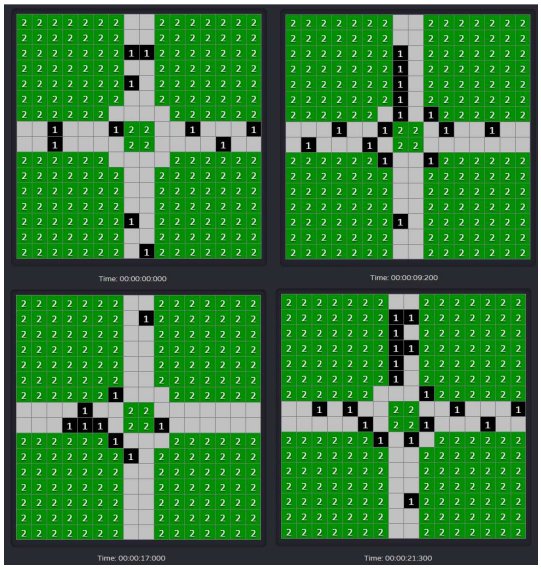

**Fig. 10.** Traffic simulation results in roundabout with yield signs

## Traffic Monitoring in Highways

In this section, we present a model for traffic monitoring on a one-way two-lane highway. The Traffic Monitor can detect unusual events. The model is as follows. We use two layers; one that models the highway and sends information to the traffic monitor and a second layer that shows the information got by the traffic monitoring system. The information that the traffic monitor gets is as follows. If within 2 time steps one unit keeps occupied, -1 is sent to the Traffic Monitor, which will see whether it is an accident or congestion as follows:

- If it receives the value -1 from unit i of both lanes at the same time step, it is congestion.
- If it only receives -1 from unit i of only one lane at the time, it is an accident.

Each vehicle has an integer velocity ranging from 0 to Vmax=5 units/time step. Assuming that the average vehicle length is 7.5m and the time step is 1 second, the Vmax is approximately 135km/h.

We model the following behaviors:

1. *Acceleration*: if the velocity v of a vehicle is lower than Vmax and the distance to the next vehicle ahead is larger than v+1, the speed is incremented.
2. *Slowing down*: if a vehicle at unit i detects the next vehicle at unit i+j (with j<=v), its speed is decremented.
3. *Changing lane*: if a vehicle at unit i detects a stopped vehicle in unit i+1, it will try to change to the other lane as long as that unit is empty and no vehicles may be able to come across at the next time step.
4. *Traffic Monitor*: if an accident occurs or congestion happens, the Traffic Monitor is able to detect the value from that exact unit.

The formal Cell-DEVS specification is as follows:

$TrafficMonitor\ AM = < X, Y, \mathrm{I}, S, \theta, \mathrm{E}, \mathrm{delay}, \mathrm{d}, \delta_{ext}, \delta_{int}, \tau, \lambda, D >$

X = Y = {-1,0,1,2,3,4,5,9}

S = {s|s∈{-1,0,1,2,3,4,5,9}}//-1 means a cell is stopped by an accident or congestion, values from 0~5 means the cell is occupied by a vehicle and the number is the vehicle's velocity and 9 means the cell is empty

I and θ are defined as above

E={(0,-5,0),(0,-4,0),(0,-3,0),(0,-2,0),(0,-1,0),(0,0,0),(0,1,0),(0,0,1),(1,-5,0),(1,-4,0),(1,-3,0),(1,-2,0),(1,-1,0),(1,0,0),(1,1,0)}

*delay*: transport.

d: 100 ms

τ: is defined by the following rules:

(0,0,0) = -1 **if** {(0,0,1) = 0}
(0,0,0) = 0 **if** {(0,0,0)!=9 and (0,1,0)!=9 and (0,1,0)!=0 and (0,0,0)!=-1}
(0,0,0) = 0 **if** {(0,0,0)!=9 and (0,1,0)=0 and (1,1,0)!=9}
…
(0,0,0) = 0 **if** {(0,0,0)!=9 and (0,1,0)=0 and (1,-5,0)!=9}
(0,0,0) = 1 **if** {(0,-1,0)=0 and (0,0,0)=9}
(0,0,0) = 1 **if** {(0,0,0)=9 and (0,-1,0)=9 and (0,-2,0)=9 and (0,-3,0)=9 and (0,-4,0)=9 and (0,-5,0)=9 and (1,0,0)=0 and (1,-1,0)!=9}
…

(0,0,0) = 9 **if** {(0,0,0)!=9 and (0,1,0)=0 and (1,0,0)=9 and (1,-1,0)=9 and (1,-2,0)=9 and (1,-3,0)=9 and (1,-4,0)=9 and (1,-5,0)=9 and (1,1,0)=9}

δint, δext , λ and D are defined using Cell-DEVS specifications.

Once the behavior of each cell is defined, the cell space is built as a Cell-DEVS coupled model:

$$TrafficMonitor\ CM = < Xlist, Ylist, \mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{n}, \{t_1, \ldots, t_n\}, \mathrm{N}, \mathrm{C}, \mathrm{B}, \mathrm{Z}, \mathrm{select} >$$

X = Y = Xlist = Ylist = ∅

n=3

$\{t_1,\ldots,t_n\} = \{2,60,2\}$ // These parameters of the model change as we change the size of the cell space.

N = {(0,-5,0),(0,-4,0),(0,-3,0),(0,-2,0),(0,-1,0),( 0,0,0),(0,1,0),(0,0,1), (1,-5,0),(1,-4,0),(1,-3,0),(1,-2,0),(1,-1,0),(1,0,0),(1,1,0)}

Z = is defined as above based on N.

select=N


**Fig. 11.** Traffic monitor simulation results

As explained above, the CELL-DEVS space has two layers, one for the highway and another for the values sent to the traffic monitoring system. In figure 11, we show some simulation results. The first layer represents the vehicles in the highway. The second layer represents the information received by the traffic monitoring system. Gray cells represent empty cells in the first layer

and no warning in the second layer. In the first layer, red cells represent vehicles. The intensity of the color represents the velocity (darker is faster). In the second row, a blue cell represent that the system gets a warning. This warning is received when a vehicle is stopped in the highway. In the simulation, we can see how at the beginning vehicles go slowly due to the presence of an accident (both lines have a stopped vehicle at the same point). After a while, this issue disappears and the traffic is more fluid. At the end of the simulation, most of the vehicles go at the highest speed and the traffic monitor does not receive incidents.

## Conclusion

We presented the use of Cell-DEVS for modeling different CA with emerging behavior for traffic. Using this methodology, new traffic rules can be tested in both simple and complex scenarios before setting new policies. Moreover, the visualization engine provides a useful and nice way to show the emergent behavior of the designed traffic rules to policy makers, who are the people responsible in decision-making.

The Cell-DEVS formalism in urban research is similar to CA but with several advantages in comparison. Cell-DEVS is more flexible than CA as it can be easily coupled with other DEVS models, and we can define timing delays for each cell, making the timing specification more expressive. We have showed how to use such Cell-DEVS formalism and the CD++ toolkit to study traffic.

## Acknowledgements

This work is partially supported by, University of Valladolid, Banco Santander and NSERC.

## References

[1] Burks, A.W.: Von Neumann's self-reproducing automata. In: Burks, A.W. (ed.) Essays on Cellular Automata, pp. 3–64. University of Illinois Press, Champaign (1970)
[2] Wainer, G.: Discrete-Event Modeling and Simulation: a Practitioner's approach. CRC Press, Taylor and Francis (2009)
[3] Zeigler, B.; Kim, T.; Praehofer, H. "Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems". Academic Press. 2000.
[4] Wainer, G., Castro, R.: A survey on the application of the Cell-DEVS formalism in cellular models. Journal of Cellular Automata 5(6), 509–524 (2010)
[5] Saadawi, H., Wainer, G.: Modeling Physical Systems Using Finite Element Cell-DEVS. Simulation Modelling Practice and Theory 15(10), 1268–1291 (2007)
[6] Wainer, G., Davidson, A.: Defining a Traffic Modeling language Using Cellular Discrete-Event abstractions. Journal of Cellular Automata 2(4), 291–343 (2007)
[7] Miller, F.P, Vandome, A.F, McBrewster, J. Cellular Automaton: Elementary Cellular Automaton, Brian's Brain, Conway's Game of Life, Langton's Ant, Rule 90, Rule 110, Rule 184, Codd's Cellular Automaton. (2009) Alpha Press.
[8] Li, Wentian: Power spectra of regular languages and cellular automata. Complex Systems 1: 107–130.(1987)
[9] Krug, J.; Spohn, H. Universality classes for deterministic surface growth. Physical Review A 38 (8): 4271–4283 (1988).