Specifying Truck Movement in Traffic Models Using Cell-DEVS

Alejandra Davidson Gabriel Wainer

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Pabellón I - Ciudad Universitaria
Buenos Aires (1428) – ARGENTINA
{ad3n,gabrielw}@dc.uba.ar
http://www.dc.uba.ar/people/proyinv/celldevs

Abstract

A specification language was defined to outline sections of cities as cell spaces. The goal is to allow the definition of complex traffic models in a simple fashion for the modeler. Once the urban section is outlined, the traffic flow is automatically set up. In this case, the language was expanded to include the behavior of trucks. The models are formally specified, avoiding a high number of errors in the developed application, and the problem solving time can be reduced.

1. Introduction

Traffic simulations are useful to test traffic policies, traffic signals, measuring the consequences of collisions or men at work, controlling pollution, avoiding traffic jams, etc. Due to the complex characteristics of these systems, a model reflecting a higher number of features can provide more accurate results. The represented behavior must be more detailed, involving the need of higher computing power.

The basic aspect of transit models is the structure chosen to represent streets, because the structure defines the kind of movements allowed. Simple models represent transit flow on one-lane roads. More complex ones allow bi-directional multi-lane roads with street intersections. These must include the exchange of vehicles between lanes and vehicles turning in the corners. Other important aspects include how to represent of different vehicles, control signals, deviations, accidents, etc.

Cellular Automata [1] is a formalism well suited to describe these problems. A model built using this paradigm is represented as a lattice of cells using discrete variables for time, space and system states. The cells in the lattice are updated according with a local rule in a simultaneous and synchronous way, using a local computing function. This function considers the state of the present cell and a finite set of nearby cells (called the neighborhood). Several works proposed to use cellular automata for traffic simulations (see [2]). Nevertheless, most of them are devoted to model simple aspects of the traffic flow.

An important constraint is that cellular automata are synchronous. This fact reduces the timing precision for the models, and compute time can be wasted when the cells are quiescent. The Cell-DEVS paradigm [3] solve these problems by describing cell spaces as discrete event models using the DEVS formalism [4]. The paradigm allows to include delay functions to have a simple definition of the timing of the cell.

This work presents the definition of ATLAS, a specification language used to define traffic models [5]. This high level language is mapped into Cell-DEVS models, thus improving the model definition and its execution precision. The work is devoted to present constructions defining the behavior of trucks, a detailed traffic behavior not existing in most cellular models.

2. Background

ATLAS is defined as a set of components allowing the definition of traffic behavior [5]. A city section is specified by a set of streets connecting two **crossings**. Each street is represented as a sequence of **segments**, representing a road section of one block of length. Every lane of each segment has the same traffic direction (one way)

This work was partially supported by ANPCYT Project 11-04460 and UBACYT Project JW10.

and a maximum allowed speed. Consequently, a two-way street is built by defining one segment for each direction.

Different sets of rules were defined for segments with one to five (or more) lanes, because the behavior of the borders is different in each case. The speed of each vehicle is represented through a delay function, using a random variable related with the speed limit allowed.

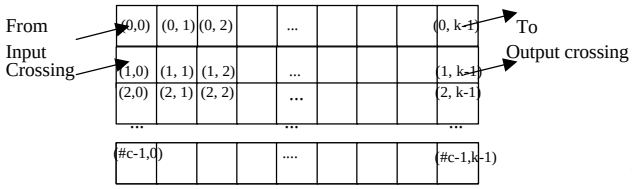

Figure 1. A segment with more than four lanes.

The crossings are represented as a ring of cells where the vehicles advance or turn to a new segment [6]. A car into the intersection has higher priority to obtain a new position in the ring than the cars out of the crossing. In order to avoid deadlocks, the cars advance continuously. Each crossing is translated into a unidimensional cellular model. The inputs and outputs of the model are obtained by analyzing the segments connected to the crossing and their direction.

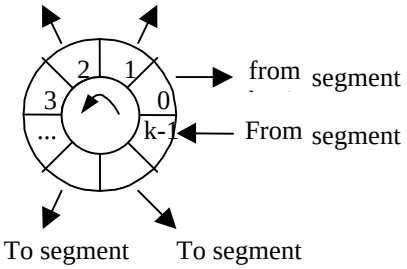

Figure 2. Crossing

These constructions are connected to an experimental framework, defined as a set of segments allowing inputs and outputs for the city section analyzed. They are defined as:

$$\text{InputSegments} = \{ s \mid s = (p1, p2, n, a, dir, max) \wedge s \in \text{Segments} \wedge [(dir = 0 \wedge (0\ v \in \mathbb{N} : (p2,v) \in \text{Crossings})) \vee (dir = 1 \wedge (0\ v \in \mathbb{N} : (p1,v) \in \text{Crossings}))] \}$$

$$\text{OutputSegments} = \{ s \mid s = (p1, p2, n, a, dir, max) \wedge s \in \text{Segments} \wedge [(dir = 0 \wedge (0\ v \in \mathbb{N} : (p1,v) \in \text{Crossings})) \vee (dir = 1 \wedge (0\ v \in \mathbb{N}: (p2,v) \in \text{Crossings}))] \}$$

For each $s \in$ InputSegments, a DEVS model is defined. Its goal is to generate vehicles that are inserted in the city section to be simulated. These models are defined by:

$$\text{Generator (\#c)} = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

Instead, the output segments $s \in$ InputSegments define a DEVS model devoted to consume vehicles and compute statistics.

$$\text{Output(\#c)} = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

In both cases, *#c* represents the number of lanes of the segment *s*.

The constructions have been mapped into DEVS and Cell-DEVS models, with the benefits of using a formal approach. Errors in the simulation can be detected by analyzing specifications, and the analysis of the underlying software system can be avoided. A DEVS model is seen as composed atomic submodels than can be combined into coupled models. A DEVS atomic model is described as:

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

Here, ***I*** is the model's interface, ***X*** is the input events set, **S** is the state set, and ***Y*** is the output events set. There are also several functions: δ_{int} manages internal transitions, δ_{ext} external transitions, λ the outputs, and **D** the elapsed time.

A DEVS coupled model is defined as:

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

Here, ***I*** is the model's interface, ***X*** is the set of input events, and ***Y*** is the set of output events. **D** is an index of components, and for each $i \in D$, $\mathbf{M_i}$ is a basic DEVS model, where $M_i = < I_i, X_i, S_i, Y_i, \delta_{inti}, \delta_{exti}, ta_i >$. $\mathbf{I_i}$ is the set of influencees of model i. For each $j \in I_i$, $\mathbf{Z_{ij}}$ is the i to j translation function.

When Cell-DEVS is used, each cell is defined as an atomic DEVS model. Transport and inertial delays allow to define timing behavior. A *transport* delay allows us to model a variable response time for each cell. Instead, *inertial* delays are preemptive: a scheduled event is executed only if the delay is consumed. Cell-DEVS atomic models can be formally specified as:

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

In this case, ***X*** represents the external input events, ***Y*** the external outputs, and ***I*** is the interface of the model. **S** is the cell state definition, and **N** is the set of input events. **Delay** defines the kind of delay for the cell, and **d** its du-

ration. Each cell uses the inputs to compute the future state using the boolean function τ. The delay allows to defer the outputs. This behavior is defined by the δ_{int}, δ_{ext}, λ and **D** functions.

A Cell-DEVS coupled model is defined by:

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

Here, **Ylist** is the output coupling list, **Xlist** is the input coupling list and ***I*** represents the interface of the model. ***X*** are the external input events and ***Y*** the external outputs. The **n** value defines the dimension of the cell space, $\{\mathbf{t_1,...,t_n}\}$ is the number of cells in each dimension, and **N** is the neighborhood set. **C** is the cell space, **B** is the set of border cells and **Z** the translation function.

ATLAS includes other complex constructions: traffic lights, railways, men at work, street holes, transit signals, parked cars, and so on. Finally, specialized behavior can been defined for certain vehicles: trucks, vans and high priority cars (ambulances, police, firefighters). The following sections are devoted to present the behavior of segments and crossings when truck movement must be analyzed.

3. Segments with trucks

The constructions presented in the previous section were redefined to allow the representation of trucks. Two different constructions are available according to the kind of traffic allowed in the street. The standard models are used in streets where heavy traffic is not allowed. Otherwise, the models presented in these sections are applied. A segment including trucks is defined by:

TruckSegments = { (p1, p2, n, a, dir, max) / p1, p2 ∈ City ∧ n, max ∈ N ∧ a, dir ∈ {0, 1} }

Here, **p1** and **p2** represent the boundaries of the segment, defined as points in City = { (x,y) / x, y ∈ Z }; **n** ∈ N, is used to define the number of lanes in the segment; **dir** ∈ {0,1}, represents the vehicle direction (dir = 1 for vehicles moving towards p2; and 0 otherwise); **a** ∈ {0, 1}, defines the shape of the segment (0 for a straight line, 1 for a curve), and **max** ∈ N, is the maximum speed allowed in the segment. Each segment will be translated into a Cell-DEVS with transport delays, defined as:

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

X = ***Y*** = N;

delay = transport; **d** = truck_sp(max);

S:

$$s = \begin{cases} 1 & \text{if there is a car in the cell;} \\ 0 & \text{if the cell is empty; and} \\ k = r \bmod 10 \wedge\ r \in [2,5] & \text{if here is a truck.} \end{cases}$$

D, λ, δ_{int} and δ_{ext} are defined by the Cell-DEVS formalism to achieve the transport delay behavior.

τ will be presented later;

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

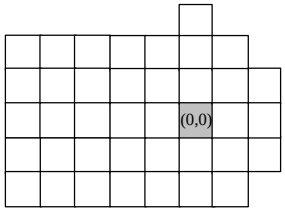

Figure 3. Cell's neighborhood.

A truck is defined by { s / s ∈ N ∧ s > 1 ∧ r = s mod 10 ∧ r ∈ [2,5] }. Each *s* represents an identifier for a truck with length *r*. The length is used to define the space needed to change between lanes. Each truck has a unique identifier (the *s* parameter) that can be used to recognize it. *Truck_sp()* is a random function returning a delay representing the speed. The value returned depends on the maximum speed allowed in the street.

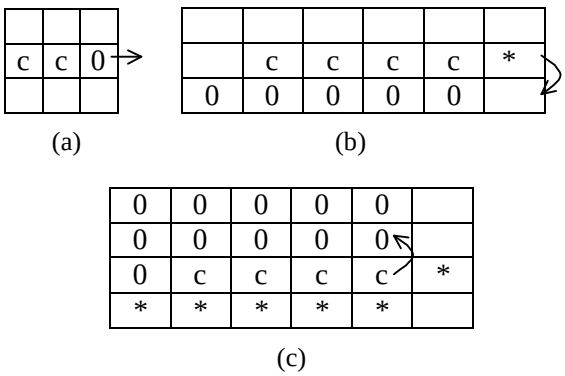

Figure 4. Truck movements (a) Straight; (b) Change to the right lane; (c) Change to the left.

The cell behavior defined by the τ function is divided in two groups. The first one is related with the truck's movement, and the second is related with car's advance. When a truck moves, several cells should change together. The first cell is in charge to choose the next movement, checking if there is space to move all the truck. The remaining cells will follow the first one.

There are three valid movements. The first one movement consists in advancing one cell in the same lane. If there is no space, it tries to move to the right (reflecting that the trucks should use the right lanes). Otherwise, it tries to go to the left.

As the segments contain trucks or cars, a policy ruling the movement of both must be defined. Any vehicle advancing straight will have a higher priority than the ones coming from the other lanes. Therefore, to make a straight movement we only need to check if there is a place in the previous cell. A truck trying to move to the right must check if there is enough place to do it. These trucks will have a higher priority than the cars trying to make the same movements. Instead, when moving to the left, the trucks should check that no vehicles are trying to occupy those cells. The neighborhood defined in the Figure 3 allows to check these movements by using the following rules:

τ(N)	N
(0,-1)	(0,0) = 0 and Truck(0,-1) and IsHead(0,-1)
(0,-1)	(0,0) = 0 and Truck(0,-1) and (0,1) = (0,-1)

Here, we represent the arrival of a truck to the origin cell from the cell in the back. *Truck(i,j)* is a macro used to verify that the position contains a part of a truck, that is, Truck(i,j) ≡ remainder((i,j),10) ∈ [2,5]. *IsHead(i,j)* is a macro that checks if the position (i,j) is the head of the truck. To do so, it verifies that the previous positions do not include part of the same truck.

The following rules represent a truck leaving the cell. In the first case, the truck leaves the cell using a straight movement. In the second one, the truck moves to the right.

τ(N)	N
0	Truck(0,0) and (0,1) = 0 and IsHead(0,0)
0	Truck(0,0) and (0,1) = 0 and (0,2) = (0,0)

In the next case, the head and the rest of the truck are also distinguished. Only the head checks if there is enough space to move (*Free_Right_Truck(0,0)*), and the rest just follow it.

τ(N)	N
0	Truck(0,0) and Free_Right_Truck(0,0) and IsHead(0,0)
0	Truck(0,0) and (-1,1) = (0,0)

Here, *Free_Right_Truck(i,j)* verifies if the right lane of the cell (i,j) is free, that is, Free_Right_Truck(i,j) ≡ ∀ k ∈ [0, remainder((i,j),10)] ⇒ (i-1,j-k) = 0.

The following rule represents a truck arriving to the origin cell from the left. The head should check that the chosen movement is to the right (*((1,1) != 0)*, and there is space to do it (*Free_Right_Truck(1,0)*).

τ(N)	N
(1,0)	(0,0) = 0 and Truck(1,0) and (1,1) != 0 and IsHead(1,0) and Free_Right_Truck(1,0)
(1,0)	(0,0) = 0 and Truck(1,0) and (0,1) = (1,0)

A truck leaving the origin cell and moving to the left is represented following. The truck's head checks if there is enough place to move to the left (*2LeftLanesFree(0,0)*). Two lanes are necessary because other trucks or cars trying to move have higher priority to do it.

τ(N)	N
0	Truck(0,0) and 2LeftLanesFree(0,0) and IsHead(0,0)
0	Truck(0,0) and (1,1) = (0,0)

Following, we represent a truck arriving to the origin cell from the right. If it is the truck's head, the movement should be checked by seeing that the truck cannot advance (*((1,1) != 0)* or turn right (*!(Free_Right_Truck(-1,0))*). It also should have enough place to do the movement (*2LeftLanesFree(-1,0)*).

τ(N)	N
(-1,0)	(0,0) = 0 and Truck(-1,0) and (-1,1) != 0 and IsHead(-1,0) and !(Free_Right_Truck(-1,0)) and 2LeftLanesFree(-1,0)
(-1,0)	(0,0) = 0 and Truck(-1,0) and (0,1) = (-1,0)

The car movements also must be defined. The first rule following represents the arrival of a car to the cell. The second one represents a car abandoning a cell, and moving forward. In these cases, the delay function is d = speed(max), where *speed* returns a delay proportional to the car speed depending on the speed limit.

τ(N)	N
1	(0,0) = 0 and (0,-1) = 1
0	(0,0) = 1 and (0,1) = 0

The next rule represents a car leaving a cell to the left. It checks that enough place is available (*((1,0) = 0 and (1,1) = 0*) and no trucks with higher priority are trying to move to the same cell (*!(Truck(2,1))*).

τ(N)	N
0	(0,0) = 1 and (1,0) = 0 and (1,1) = 0 and !(Truck(2,1))

The following rule represents a car arriving to the origin cell from the right. The car should not be able to move forward (*((-1,0) != 0*), it should have enough space to move (*((0,-1) = 0 and (0,0) = 0*), and a truck should not be able to arrive to the same position (*!(Truck(1,0))*).

τ(N)	N
1	(0,0) = and (-1,-1)=1 and (-1,0) !=0 and (0,-1) =0 and !(Truck(1,0))

When a car moves to the right, we must check that enough space to move is available. In addition, no trucks (with higher priority) or cars must be trying to reach same position (*((-2,0) = 0 or (-2,1) = 0*).

τ(N)	N
0	(0,0) = 1 and (-1,0) = 0 and (-1,1) = 0 and !(Truck(0,1)) and ((-2,0) = 0 or (-2,1) = 0)

Finally, a car can arrive from the left. We must check there is no space to advance, the car cannot stay in its own lane, and it cannot move to the left. Besides, it checks if there is a truck moving to the origin cell.

τ(N)	N
1	(0,0) = 0 and (1,-1) = 1 and (1,0) != 0 and !(Truck(1,0)) and (0,-1) = 0 and ((2,-1) != 0 or (2,0) != 0 or Truck(3,0))

After defining the individual behavior of each cell in the model, a Cell-DEVS coupled model should be built for the segment. In this case, the coupled model is defined by:

$$SL(k, max, \#s) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z >$$

Ylist = { (i,0) / i ∈ [0, #s] } ∪ { (i,k-1) / i ∈ [0, #s] }

Xlist = { (i,0) / i ∈ [0, #s] } ∪ { (i,k-1) / i ∈ [0, #s] }

I = <P^x, P^y>, with P^x = {<$X_{\eta+1}$(i,0), ***N***> / i ∈ [0, #s]} ∪ {<$X_{\eta+1}$(i,k-1), ***N***> / i ∈ [0, #s]} and P^y = {<$Y_{\eta+1}$(i,0), ***N***> / i ∈ [0, #s]} ∪ {<$Y_{\eta+1}$(i,k-1), ***N***> / i ∈ [0, #s]}.

X = ***Y*** = N;

n = 2; $\mathbf{t_1}$ = #s; $\mathbf{t_2}$ = k;

C = { C_{ij} / i ∈ [0, #s-1] ∧ j ∈ [0, k-1] };

B = { (i,j) / ∈ [0, #s-1] ∧ j ∈ [0, 5] } ∪ { (i,j) / i ∈ [0, #s-1] ∧ j ∈ [k-2, k-1] }; and

Z is built by following the definition for Cell-DEVS models.

Here, SL(k, max, #s) represents a segment of *#s* lanes of length *k* each with speed level *max*. Variable *k* is computed using *Cells-Nr(t)*, a function that takes the length of the segment and the size of each cell.

The following input/output ports are used:

Name	Comment
x-c-vehicle	A vehicle leaves a crossing to a segment.
x-c-room	There is enough room in the crossing so a vehicle can advance from the segment.
y-c-room	There is enough room in the segment so a vehicle can leave the crossing.
y-c-vehicle	A vehicle in the segment is trying to cross.

The border cells of the coupled model are connected with the crossings, therefore, their behavior is different from the rest of the cell space. The column 0 will be devoted to receive trucks from the crossing. The trucks in a crossing are represented using only one cell. Hence, when they return to a segment their full length must be reconstructed. This rule represents a car arriving to the origin cell from the crossing:

τ(N)	N
1	(0,0) = 0 and x-c-vehicle = 1

The following rule represents the arrival of a truck from the crossing:

τ(N)	N
x-c-vehicle	(0,0) = 0 and IsTruck(x-c-vehicle)

The following rules will represent that the truck size expands when it leaves the crossing:

τ(N)	N
0	Truck(0,0) and (0,1) = 0 and CompleteTruck(0,0) send(0,y-c-room)
0	Truck(0,0) and (0,1) = 0 and !CompleteTruck(0,0) and !IsHead_TC1(0,0)) send((0,0),y-c-room)
(0,1)	(0,0) = 0 and Truck(0,1) and !CompleteTruck(0,1) send((0,1),y-c-room)
0	Truck(0,0) and (0,1) = 0 and IsHead_TC1(0,0) send((0,0),y-c-room)

Here, *IsTruck(k)* verifies that a value passed as parameter represents a truck, that is, IsTruck(k) ≡ remain-

der(k,10) ∈ [2,5]. *CompleteTruck(i,j)* is used to see if the truck has been generated completely.

The first rule represents that the truck has been generated completely. Consequently, the crossing should know that it is ready to receive another vehicle. This is done sending a 0 value to the crossing. The second rule represents that part of the truck advanced to the following cell, but the truck is still being reconstructed. Therefore, the cell in the crossing is busy. The third rule represents the generation of a new part of the truck in the origin cell. The last rule is equivalent to the second, but considering that the origin cell contains the truck's head. Therefore, its output must be delayed (in the other rules, 0 delays are used).

The following rules specify the size reduction of a truck arriving to a crossing:

τ(N)	N
(0,-1)	(0,0)=0 and Truck(0,-1) and CompleteTruck(0,-1) send(0,y-c-vehicle)
0	Truck(0,0) and x-c-room = 0 send((0,0),y-c-vehicle)
0	(0,0)=0 and Truck(0,-1) and !CompleteTruck(0,-1) send(0,y-c-vehicle)

The first rule represents the arrival of a truck to a crossing when there is enough room to cross. The value representing the truck's number is transmitted only if the cell contains the head of the truck. The second rule represents a truck arrived to the crossing. The third rule represents the input of the remaining parts of the truck.

Finally, the following rules are related with the interchange of cars. The next one is in charge of sending a car to the crossing:

τ(N)	N
0	(0,0) = 1 and x-c-room = 0 send(1,y-c-vehicle)

4. Crossings with trucks

When trucks are allowed in the crossing, the following construction is used:

$$\text{TruckXings} = \{ (c, maxc) / maxc \in \mathbb{N} \wedge \exists\, t, t' \in (\text{TruckSegments} \cup \text{Segments}) \wedge t = (p1, p2, n, a, dir, max) \wedge t' = (p1', p2', n', a', dir', max') \wedge t \neq t' \wedge (p1 = c \vee p2 = c) \wedge (p1' = c \vee p2' = c) \}$$

This set is defined as points in a bidimensional space, representing the places where two or more segments are crossed. It is built using Segments and TruckSegments sets. These crossings must recognize the segments allowed to receive trucks. The standard binary crossings are used for car crossings. Instead, if one of the *s* segments belongs to TruckSegments, the crossing is defined as part of TruckXings.

Each crossing (c, maxc) ∈ TruckXings is defined as a one-dimensional Cell-DEVS with transport delays:

$$C_{0j} = < I, X, S, Y, N, \delta_{int}, \delta_{ext}, delay, d, \tau, \lambda, D >$$

I = < η, P^x, P^y>, with η = 3; P^x = { (X_1, ***N***), (X_2, ***N***), (X_3, ***N***) }; P^y = { (Y_1, ***N***), (Y_2, ***N***), (Y_3, ***N***) }.
X = ***Y*** = N;
S:

$$s = \begin{cases} 1 & \text{if there is a car in the cell;} \\ 0 & \text{if the cell is empty;} \\ k = r \bmod 10 \wedge r \in [2,5] & \text{if there is a truck.} \end{cases}$$

N = { (0,-1), (0,0), (0,1) };
delay = transport; **d** = speed(maxc);
D, λ, δ_{int} and δ_{ext} are defined by the Cell-DEVS formalism with transport delays.
The function τ will be defined later.

The coupled model corresponding to the crossing (c, maxc) is defined by:

$$\text{TruckXings}(k, In, Out, Out\_Cars, 1, maxc) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z >$$

Ylist = { (0,i) / i ∈ [0, k]};
Xlist = { (0,i) / i ∈ [0, k] };
I = <P^x, P^y>, with P^x = {<$X_{\eta+1}$(0,i), binary> / i ∈ [0, k]}, P^y = {<$Y_{\eta+1}$(0,i), binary> / i ∈ [0, k]}.
X = N;
Y = N;
n = 1;
$\mathbf{t_1}$ = k;
C = { C_{0j} / j ∈ [0, k-1] };
B = {∅};
Z is built using the specification of Cell-DEVS; and

This is a crossing of *k* cells, with a maximum speed of *maxc*. The positions of In are the inputs to the crossing, and Out/Out_cars are the outputs. Here, Out_cars represents the set of segment that cannot receive trucks. These sets are obtained by computing

{I, O} = Ports_In_Out((c, maxc), Segments ∪ Truck-Segments)

The function *Ports_In_Out* takes the specification of the segments connected to the crossing and their direction. The result of the {I, O} sets define which cells are connected with each input/output segment.

The following ports are used:

Name	Comment
x-s-vehicle	A vehicle is trying to get into the crossing.
x-s-room	There is a place to leave the crossing.
y-s-room	The crossing has space for a vehicle.
y-s-vehicle	A vehicle is leaving the crossing.

Using these definitions, τ will be specified. A different behavior should be provided for input and output cells. Each of them will be explained with detail in the following sections.

4.1. Definition of the output cells

The output cells of the crossing will behave according to the segment to which they are coupled. The output cells allowing cars or trucks are defined following:

τ(N)	N
1	(0,0) = 0 and (0,-1) = 1 and [x-s-room = 1 or IsTruck(x-s-room) or (x-s-room = 0 and random < p_{out})] send(0,y-s-vehicle) /* Arrival of a car that will stay in the crossing */
(0,-1)	(0,0) = 0 and Truck(0,-1) and [x-s-room = 1 or IsTruck(x-s-room) or (x-s-room = 0 and random < p_{out})] send(0,y-s-vehicle) /* Arrival of a truck staying in the crossing */
0	(0,0) = 0 and (0,-1) = 1 and x-s-room = 0 and random ≥ p_{out} send(1,y-s-vehicle) /* Arrival of a car that will leave the crossing */
0	(0,0) = 0 and Truck(0,-1) and x-s-room = 0 and random ≥ p_{out} send((0,-1), y-s-vehicle) /* Arrival of a truck that will leave crossing */
0	(0,0) = 1 and (0,1) = 0 send(0,y-s-vehicle)
0	Truck(0,0) and (0,1) = 0 send(0,y-s-vehicle)
(0,0)	TRUE send(0,y-s-vehicle) /* Otherwise, the present state is kept */

Here, p_{out} is a constant that represents the probability of vehicle leaving the crossing. A vehicle can leave the crossing if there is enough space in the segment and a random value is higher than $p_{out.}$

A truck passing through cells connected to segments not allowing trucks will keep advancing. The behavior for these cells is defined by:

τ(N)	N
1	(0,0) = 0 and (0,-1) = 1 and (x-s-room = 1 or (x-s-room = 0 and random < p_{out})) send(0,y-s-vehicle) /* Arrival of a car that will stay in the crossing */
(0,-1)	(0,0) = 0 and Truck(0,-1) send(0,y-s-vehicle) /* Arrival of a truck that will stay in the crossing */
0	(0,0) = 0 and (0,-1) = 1 and x-s-room = 0 and random ≥ p_{out} send(1,y-s-vehicle) /* Arrival of a car that will leave the crossing */
0	(0,0) = 1 and (0,1) = 0 send(0,y-s-vehicle)
0	Truck(0,0) and (0,1) = 0 send(0,y-s-vehicle)
(0,0)	TRUE send(0,y-s-vehicle) /* Otherwise, the present state is kept */

These rules represent that, when there is space in the segment and a random value is greater than p_{out}, the cars can leave the crossing. When a truck arrives to these cells, it just advances to the next cell if it is empty. A truck cannot it leave the crossing through these cells.

4.2. Definition of the input cells

The local computing function of the input cells is defined by:

τ(N)	N
1	(0,0) = 0 and (0,-1) = 1 send(1, y-s-room)
1	(0,0)=0 and x-s-vehicle = 1 and (0,-1)=0 send(1, y-s-room)
(0,-1)	(0,0) = 0 and Truck(0,-1) ; send(1, y-s-room)
x-s-vehicle	(0,0) = 0 and IsTruck(x-s-vehicle) and (0,-1)=0 send(1, y-s-room)
0	(0,0) = 1 and (0,1) = 0 and (0,-1) = 0 send(0, y-s-room) /* No vehicle with priority is in the crossing */
0	Truck(0,0) and (0,1) = 0 and (0,-1) = 0 send(0, y-s-room) /* No vehicle with priority is in the crossing */
0	(0,0) = 1 and (0,1) = 0 and ((0,-1) = 1 or

	Truck(0,-1)) send(1, y-s-room) /* No vehicle with priority is in the crossing */
0	Truck(0,0) and (0,1) = 0 and ((0,-1) = 1 or Truck(0,-1)) send(1, y-s-room) /* No vehicle with priority is in the crossing */
(0,0)	TRUE /* Otherwise, keep the previous state */

The ports should be updated explicitly, because the new state for each of the cells is not transmitted. In the third and fourth rules, the new state represents the existence of a truck. Then, if the coupled segment is binary, it will not accept trucks. Therefore, a value of 1, representing that the cell is busy is sent to the segment. The segment will wait the cell to be empty, which will be done by sending a 0 value through the output port. In this way, if there is a vehicle with priority in the crossing, the cell sends a 1. In this way, a new arriving vehicle knows that it must stop before the crossing.

4.3. Coupling between Segments and Crossings

A crossing c = (p, maxc) influences the segments s to which it is connected, that is,

$I_c = \{ M_s / s \in (\text{Segments} \cup \text{TruckSegments}) \wedge s = (p1, p2, n, a, dir, max) \wedge (p1 = p \text{ or } p2 = p) \}$

Therefore, a segment s influences to the two crossings in its border:

$I_s = \{M_{c1}\} \cup \{M_{c2}\}$, if s = (p1, p2, n, a, dir, max) and
$(\exists v1, v2 \in \mathbb{N} : c1, c2 \in (\text{Crossings} \cup \text{TruckXings}) \wedge c1 = (p1, v1) \wedge c2 = (p2, v2))$

These ports are coupled by defining the Z function. To define the cells in the crossing and the segment that will be connected to the {I, O} sets computed previously.

As the coupling is done in the first (0) and last (k-1) cells, for each (s,i) ∈ I, s = (p1, p2, n, a, dir, max) we must know the number of cells in the segment. These values are obtained computing the length of the segment (using p1 and p2) and dividing the result by the size of each cell. In this case, Z is defined by:

$Z_{sc} : Y_{\eta+1}(j, k-1)_s \rightarrow X_{\eta+1}(0,i+j)_c, \forall (j \in \mathbb{N}, j \in [0, n-1])$
$Z_{ct} : Y_{\eta+1}(0,i+j)_c \rightarrow X_{\eta+1}(j, k-1)_s, \forall (j \in \mathbb{N}, j \in [0, n-1])$

Instead, for each (s,i) ∈ O with s = (p1, p2, n, a, dir, max), Z is defined by:

$Z_{cs} : Y_{\eta+1}(0,j+i)_c \rightarrow X_{\eta+1}(n-1-j, 0)_s, \forall (j \in \mathbb{N}, j \in [0, n-1])$
$Z_{sc} : Y_{\eta+1}(n-1-j, 0)_s \rightarrow X_{\eta+1}(0,j+i)_c, \forall (j \in \mathbb{N}, j \in [0, n-1])$

5. Conclusion

This article was devoted to show part of a specification language used to define sections of cities as cell spaces. The work was focused into the definition of truck movements in the cell spaces. Each construction of the specification language was translated into Cell-DEVS models.

This approach provides an application oriented specification language, which will allow the definition of complex traffic behavior using simple rules. The models are formally specified, avoiding a high number of errors in the developed application. The problem solving time is highly reduced, allowing the analysis complex behavior in the traffic, and providing new solutions.

The use of Cell-DEVS not only provided a formal approach but also a discrete events basis, allowing to improve the execution performance. Whenever a cell is not active (due to lack of vehicles or bottlenecks), it will not affect the rest of the simulation. Different behavior with changing complexity can be achieved by using different kinds of constructions. In this way, non homogeneous cellular models can be easily constructed, adding complexity only where is needed.

At present, the specification language is being implemented. In addition, a graphical user interface is being defined, allowing easy definition of the models. Finally, efficient execution of the models is being considered by means of parallel execution of the Cell-DEVS models.

References

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

[2] Davidson, A.; Díaz, A.; Vázquez, V. and Wainer, G. "A comparative study of cellular automata application for simulation of urban traffic models". *Technical Report 99-005, Departamento de Computación, FCEN/UBA.* Submitted to publication. 1999.

[3] Wainer, G. and Giambiasi, N. "Specification, modeling and simulation of timed Cell-DEVS models". *Technical Report 97-007, Departamento de Computación, FCEN/UBA. Submitted to publication.* 1998.

[4] Zeigler, B. *Multifaceted Modelling and discrete event simulation.* Academic Press, 1984.

[5] Davidson, A. and Wainer, G. "Definition of a specification language for urban traffic simulation using the Cell-DEVS formalism" (in Spanish). *Technical Report 99-003, Departamento de Computación, FCEN/UBA.* 1999.

[6] Chopard, B.; Dupuis, A.; Luthi, P. “A Cellular Automata Model for Urban Traffic and its applications to the city of Genoa”. *Proceedings of Traffic and Granular Flow*. 1997.