Application of the ATLAS language in models of urban traffic

Andrea Díaz Verónica Vázquez

Departamento de Computación
FCEN – Universidad de Buenos Aires
Planta Baja. Pabellón I.
Ciudad Universitaria (1428)
Buenos Aires. Argentina.

Gabriel Wainer

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

E-mail: gwainer@sce.carleton.ca

Abstract

ATLAS is a specification language defined to outline city sections to model and simulate traffic flow. Streets are characterized by their size, direction, number of lanes, etc. Once the urban section is outlined, the constructions are translated into Cell-DEVS models, and the traffic flow is automatically set up. As modelers can focus in the problem to solve, development times for a simulation can be highly reduced. We present an example of application of the specification language to solve specific problems.

1. Introduction

The use of simulation has been gaining popularity for urban traffic analysis and control. Modelling and simulation of traffic has been used to improve traffic control, measure the consequences of collisions, avoid pollution, traffic jams, etc.

ATLAS (Advanced Traffic LAnguage Specifications) is a high level specification language defined to represent city sections as cell spaces. It is focused to analyze detailed behavior of traffic (microsimulations) and it is not intended to model traffic flow in the large. The idea is to allow elaborate study of flow according with the shape of a city section and its traffic attributes. A city section can be easily described, including definitions for traffic signs, traffic lights, etc. Once a section is defined, the traffic movement is automatically created. Therefore, a modeler can concentrate in the problem to solve, instead of being in charge of defining a complex simulation.

The constructions defined in this language are mapped into DEVS and Cell-DEVS models, providing the benefits of a formal approach [1, 2]. Cell-DEVS [3] was proposed to describe cell spaces as DEVS models with timing delays [4]. Using Cell-DEVS, a cellular model can be described as a discrete event model. Transport and inertial delays allow timing accurate description, improving the definition of the models using explicit delays. Here, we will show the application of ATLAS, focusing an application example.

2. ATLAS Constructions

ATLAS allows to represent the structure of a city section defined by a set of streets connected by crossings. The language constructions define a static view of the model, and they are considered to be built as grids composed of cells. The main constructions are:

. **Segments**: they represent sections between two corners. Every lane in a given segment has the same direction (one way segments) and a maximum speed. They are specified as: Segments = $\{ (p1, p2, n, a, dir, max) / p1, p2 \in City \wedge n, max \in \mathbb{N} \wedge a, dir \in \{0,1\} \}$, where **p1** and **p2** represent the boundaries of each segment (City = $\{ (x,y) / x, y \in \boldsymbol{R} \}$), **n** is the number of lanes, and **dir** represents the vehicle direction. The **a** parameter defines the shape of the segment (straight or curve, allowing to define the city shape precisely, and to include the exact number of cells), and **max** is the maximum speed allowed.

. **Crossings**: they are points in the plane where several segments intersect. They are specified as: Crossings = $\{ (c, max) / c \in City \wedge max \in \mathbb{N} \wedge \exists\, s, s' \in Segments \wedge s = (p1, p2, n, a, dir, max) \wedge s' = (p1', p2', n', a', dir', max') \wedge s \neq s' \wedge (p1 = c \vee p2 = c) \wedge (p1' = c \vee p2' = c) \}$. Crossings are built as a ring of cells with moving vehicles [5]. A

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

car in the crossing has higher priority to obtain a position into the ring than the cars out of the crossing.

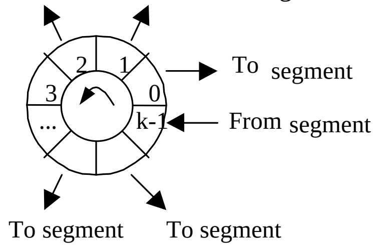

Figure 1. A crossing.

. **Traffic lights**: crossings with traffic lights are defined as: TLCrossings = { c / c ∈ Crossings }. Here, *c ∈ TLCrossings* represents several models representing the traffic lights in a corner and the corresponding controller. Each of these models is associated with a crossing input. It sends a color value related with the traffic light to the corresponding segment in the intersection.

. **Railways**: they are built as a sequence of level crossings overlapped with the city segments. The railway network is defined by: RailNet = { (Station, Rail) / Station is a model, Rail ∈ RailTrack }, where RailTrack = { (s, δ, seq) / s ∈ Segments ∧ δ ∈ ℕ ∧ seq ∈ ℕ }. Here, *RailNet* represents a set of stations connected to railways, thus defining a part of the railway network. *Railtrack* associates a level crossing with other existing constructions in the city section. Each element identifies the segment that is crossed (**s**) and the distance to the railway from the beginning of the section (δ). Finally, a sequence number (**seq**) is assigned to each level crossing, defining its position in the *RailTrack*.

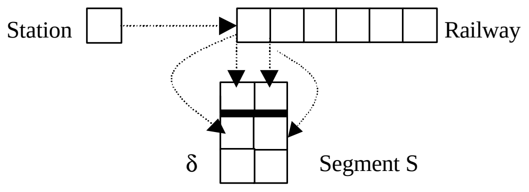

Figure 2. Level crossing definition.

. **Men at work**: they are specified as: Jobsite = { (s, ni, δ, #n) / s ∈ Segments ∧ s = (c1, c2, n, a, dir, max) ∧ ni ∈ [0, n-1] ∧ δ ∈ ℕ ∧ #n ∈ [1, n+1-ni] ∧ #n ≡ 1 mod 2 }. Here, each (*s*, *ni*, *d*, *#n*) ∈ *Jobsite* is related with a segment where the construction works are being done. It includes the first lane affected (**ni**), the distance between the center of the jobsite and the beginning of the segment (δ), and the number of lanes occupied by the work (**#n**). These values are used to define a rhombus over the segment where the vehicles cannot advance.

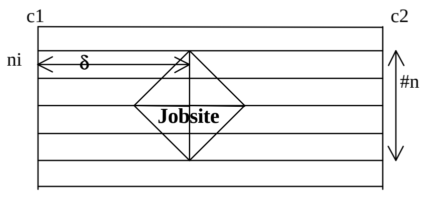

Figure 3. Segment with men at work.

. **Traffic signs**: they are defined by: Control = { (s, t, δ) / s ∈ Segments ∧ δ ∈ ℕ ∧ t ∈ {bump, depression, school, pedestrian crossing, stop, others} }. Each tuple here identifies the segment where the traffic sign is used, the kind of signal, and the distance up to it from the beginning of the segment. An extension of this construction allows us to define Potholes, whose size is one cell.

. **Truck segments**: segments in which the traffic of trucks is allowed are defined as: TruckSegments = { (p1, p2, n, a, dir, max) / p1, p2 ∈ City ∧ n, max ∈ ℕ ∧ a, dir ∈ {0, 1} } whose components are the same to those defined earlier. The idea here is to extend the behavior of standard traffic to include large size vehicles.

. **Truck crossings**: when trucks are allowed in a crossing, the following construction should be used instead of the standard crossings: TruckXings = { (c, maxc) / maxc ∈ ℕ ∧ ∃ s ,s' ∈ (TruckSegments ∪ Segments) ∧ s = (p1, p2, n, a, dir, max) ∧ s' = (p1', p2', n', a', dir', max') ∧ s ≠ s' ∧ (p1 = c ∨ p2 = c) ∧ (p1' = c ∨ p2' = c) }. These models represent points in the plane the places where several segments joins, and at least one of them should include trucks.

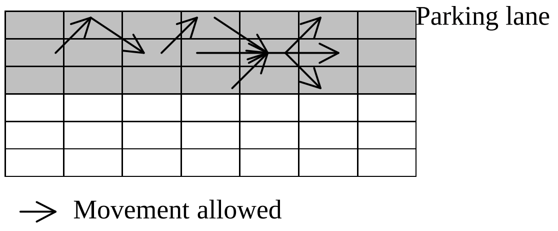

Figure 4. Parking segments.

. **Parking**: border cells in a segment can be used for parking. They are defined as: Parking = { (s, n1) / s ∈ Segments ∧ n1 ∈ {0,1} ∧ s = (c1, c2, n, a, dir, max) ∧ n > 1 }. Every pair (s, n1) identifies the segment and the lane where car parking is allowed. If n1 = 0, the cars park on the left lane. If n1 = 1, the right lane is used (lane n-1).

. **Experimental frameworks**: experimental framework constructions are defined as segments that provide inputs and outputs to the city section to be studied. They are defined as:

InputSegments = { s / s = (p1, p2, n, a, dir, max) ∧ s ∈ Segments ∧ [(dir = 0 ∧ (∃ v ∈ N : (p2,v) ∈ Crossings)) ∨ (dir = 1 ∧ (∃ v ∈ N : (p1,v) ∈ Crossings))] }

OutputSegments = { s / s = (p1, p2, n, a, dir, max) ∧ s ∈ Segments ∧ [(dir = 0 ∧ (∃ v ∈ N : (p1,v) ∈ Crossings)) ∨ (dir =1 ∧ (∃ v ∈ N: (p2,v) ∈ Crossings))] }

3. Applying ATLAS to define a city section

The following figure shows a section of Buenos Aires. It is a part of residential neighborhood, where the traffic flow is non-significant, even in the peak hours. Nevertheless, the complexity in the city trace of the area is so high that traffic jams occur frequently.

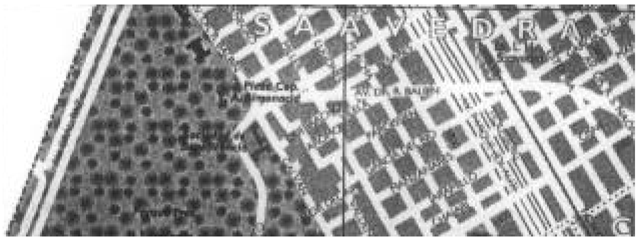

Figure 5. A section of Buenos Aires.

The area includes a park, a railway, several one way streets, dead ends, and avenues (one with five lanes in each direction). In several of these streets, parking is allowed, while in others it is forbidden. Heavy traffic is allowed in avenues. Figure 6 shows a sketch of this area, labeling the segments and crossings.

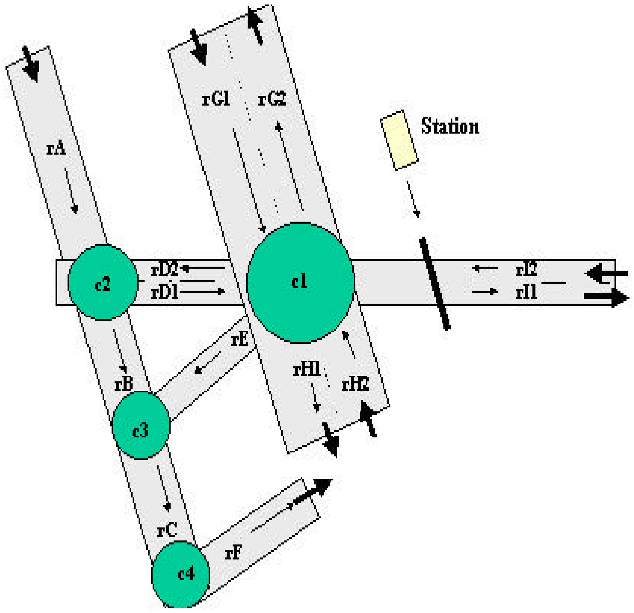

Figure 6. Sketch of the city section.

These components can be defined in ATLAS as:

Segments = { rA, rB, rC, rD1, rD2, rE, rF, rG1, rG2, rH1, rH2, rI1, rI2 }, where:

rA=[(0,0), (0,130), 1, 40, 0, 1]
rB=[(0,130), (0,200), 1, 40, 0, 1]
rC=[(0,200), (0,300), 1, 40, 0, 1]
rD1=[(0,130),(100,200),2,60,0, 1]
rD2=[(0,130),(100,200), 2, 60, 0, 0]
rE=[(0,200), (100,200), 1, 40, 0, 0]
rF=[(0,300), (100,300), 1, 40, 0, 1]
rG1=[(100,0),(100,200),4,60,0, 1]
rG2=[(100,0),(100,200),4,60,0, 0]
rH1=[(100,200),(100,300),2,60,0,1]
rH2=[(100,200),(100,300),2,60,0,0]
rI1=[(100,200),(200,280),2,60,0,1]
rI2=[(100,200),(200,280), 2, 60, 0, 0]

Crossings = { c1, c2, c3, c4 }, where c1 = ((100,200); 30), c2 = ((0,130); 30), c3 =((0,200); 20), c4 = ((0,300); 30)
RailNet = { (Station1, Rail1) }, where Rail1 = { (rI1, 90, 1), (rI2, 10, 1), ... }
TLCrossings = { c1, c2 }
Control = {(rA,school,20), (rC,stop,95), (rE,PedXing,95)}
Pothole = { (rA, 0, 10) }
TruckSegments={rD1, rD2, rG1, rG2, rH1, rH2, rI1, rI2}
TruckXings = {c2, c6, ... }
Parking = {(rG1,1) }
InputSegments = { rA, rC1, rH2, rI2 }
OutputSegments = { rH1, rI1, rF, rC2 }

The specification considers the plane starting in the segment *rA*, a one-way/one-lane segment. Its maximum speed is 40 km/h, and it is a straight line. Segments *rB* and *rC* are continuations of this segment. Segment *rD* is a two-way segment (therefore, *rD1* and *rD2* are defined). The maximum speed allowed is 60 km/h. Segment *rI* is the continuation of this segment. Finally, *rG* is a two-way segment with 4 lanes in each way. The maximum speed is also 60 km/h. As we can see, trucks are allowed in segments *rD*, *rI*, *rG* and *rH* (both ways). The crossing specifications show the position and maximum speed allowed for each of them The railway construction shows that a crossing level intersects the segment *rI*. Segment *rI1* is cut 10 m from the crossing, and *rI2*, 90 m from the next crossing. Finally, we show the definition of a pothole, several traffic signs, and a parking lane. Finally, we specify the model's experimental framework.

4. The DEVS modelling paradigm

As explained earlier, ATLAS specifications are translated into DEVS and Cell-DEVS. A real system modeled using the DEVS paradigm [4] can be described as composed of atomic or coupled submodels. A DEVS atomic model is described as:

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

The interface ***I*** includes input/output ports to communicate with other models. Input external events ***X*** are received in input ports. The model executes the external transition function δ_{ext} under such inputs. Each state has an associated duration time **D**. When this time is consumed, the internal transition function δ_{int} is activated to produce internal state changes. The internal state **S** can be used to provide model outputs ***Y*** sent through the output ports. They are sent by the output function λ, which executes before the internal transition.

A DEVS coupled model is defined as:

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

Each coupled model consists of a set **D** of basic models M_i connected through the input/output ports. The influencees I_i of a model will determine to which models one send the outputs. The translation function Z_{ij} is in charge of translating outputs of a model into inputs for the others. To do so, an index of influencees is created for each model (I_i). For every *j* in this index, outputs of the model M_i are connected to inputs in the model M_j.

Cell-DEVS allows to define cellular models that can be integrated with other DEVS. Here, each cell of a space is defined as an atomic DEVS. Transport and inertial delays define timing behavior of each cell in an explicit and simple fashion. 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 are specified as:

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

Each cell will use the **N** inputs to compute the future state **S** using the function τ. The outputs of a cell are transmitted after the consumption of the delay function associated with the cell. **Delay** defines the kind of delay for the cell, and **d** its duration. The outputs usually include the execution results of the local computing functions. This behavior is defined by the δ_{int}, δ_{ext}, λ and **D** functions.

A Cell-DEVS coupled model is defined by:

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

A cell space **C** defined by this specification is a coupled model composed by an array of atomic cells ($\{t_1 \times ... \times t_n\}$). Each of them is connected to the cells defined by the neighborhood **N**. As the cell space is finite, the borders **B** should have a different behavior than the remaining cells. Otherwise, the space is wrapped, meaning that cells in a border are connected with those in the opposite one. The **Z** function allows one to define the internal and external coupling of cells in the model. This function translates the outputs of output port *m* in cell *Cij* into values for the *m* input port of cell *Ckl*. The input/output coupling lists can be used to transfer data with other models.

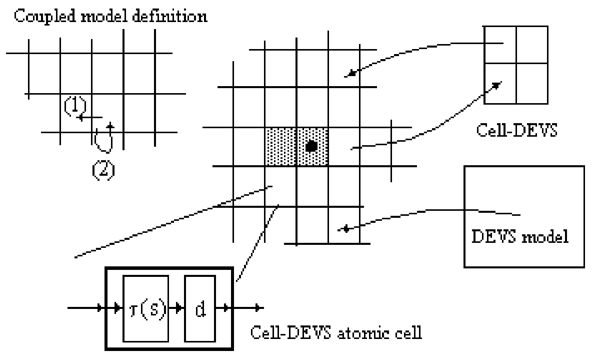

Figure 7. Informal definition of Cell-DEVS.

These formal specifications were considered to build the **CD++** tool [6]. This tool is devoted to specify DEVS and Cell-DEVS models, which can be later simulated. The models built using this tool follows the formal specifications described in this section. A specification language allows describing the behavior of each cell in Cell-DEVS, and to define coupled models. Coupled models are built by defining their size, neighborhood and borders information. Then, the cell behavior is described using rules with the following syntax: VALUE DELAY { CONDITION }. The rule semantics is that if the *CONDITION* is satisfied, the state of the cell will change to the designated *VALUE* after a *DELAY* . If the condition is not valid, the following rule is evaluated. A wide range of functions and operators can be used to define the rules, some of which will be described in the following section.

5. Translating ATLAS into Cell-DEVS

Recalling Section 3, we intend to model a city section including several segments and crossings. This section will be devoted to show the translation of these constructions into Cell-DEVS models. We introduce some of the simpler models with the goal of showing the translation procedures; the remaining are translated using similar techniques. The definition of these mechanisms were presented in [1, 2], and details about the remaining models in this example can be found in [7].

5.1. Segment definition

Segment rB is a one lane road that was specified as rB = [(0,130),(0,200),1,0,1,40]. This model is mapped into a one-dimensional Cell-DEVS with transport delays.

The first step of the translation procedure is to obtain the number of cells in the segment. This value is computed as the distance between the border points (0, 130) and (0, 200). Each cell has a fixed size of 7.5 m, representing the length of a car [5]. Hence, the size of this model can be computed as:

$$k = \left\lceil \sqrt{|x1-x2|^2 + |y1-y2|^2} \Big/ cell\_size \right\rceil = \left\lceil \sqrt{0+70^2} / 7.5 \right\rceil = 10$$

This parameter is combined with the rest defined by the construction, allows us to build the following Cell-DEVS coupled model:

rB(k=10, max=40 Km/h) = < Xlist, Ylist, I, X, Y, n, $\{t_1,...,t_n\}$, η, N, C, B, Z >
= < {(0,0),(0,9)}, {(0,0),(0,9)}, I, {0,1}, {0,1}, 1, t_1=10, 3, {(0,-1),(0,0),(0,1)}, C, {(0,0),(0,9)}, Z >

The model input/output interface is composed by the cells (0,0) and (0,9), and each of them have one input and one output port. Here **I** = <P^x, P^y>, with P^x = {<$X_{\eta+1}$(0,0), binary>, <$X_{\eta+1}$(0,9),binary>}, P^y ={<$Y_{\eta+1}$(0,0),binary>, <$Y_{\eta+1}$(0,9), binary>} (we will use the following notation: $X_{\eta+1}$(0,0): x-c-car; $X_{\eta+1}$(0,9): x-c-room; $Y_{\eta+1}$(0,0): y-c-room, $Y_{\eta+1}$(0,9): y-c-car). Then, the specification includes spatial information (we are defining a 1-dimensional Cell-DEVS with 10 cells of length, as explained earlier).

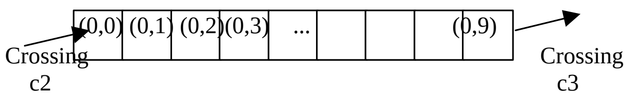

Figure 8. Shape of the segment rB

The model uses binary values to represent the existence of a car. The neighborhood includes the two close cells. We define the borders, in this case, the cells (0,0) and (0,9). Each of them is in charge interchanging information with the crossings. Finally, The C and Z sets are built using the formal definitions for Cell-DEVS models.

The border cells must behave in a different way than the rest. We must change cell (0,0) neighborhood to η= 2; **N** = { (0,0), (0,1) }. Cell (0,9) is now η = 2, **N** = { (0,-1), (0,0) } and **delay** = inertial. The cell (0,0) is connected to a cell in the crossing using two ports: one to see if a new car is passing through the crossing, and the other to inform the crossing the present value of the border. The case is similar for the other border cell. In both cases, the local computing function should provide a behavior representing the reception of new vehicles, and vehicles leaving the segments, respectively.

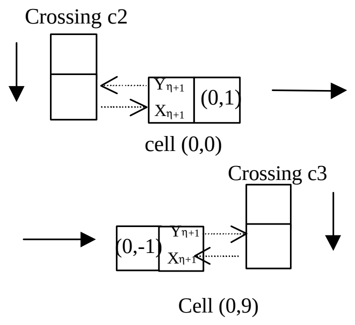

Figure 9. Connections in (0,0) and (0,9)

Every other cell in this space is defined as:

C_{0j} = < I, X, S, Y, N, δ_{int}, δ_{ext}, delay, d, τ, λ, D >
= < I, {0,1}, S, {0,1}, {(0,-1),(0,0),(0,1)}, δ_{int}, δ_{ext}, transport, speed(40Km/h), τ, λ, D >

Each cell can receive three inputs that will be used by the local transition function τ to represent the car advance. We use a transport delay to model the time a vehicle takes to abandon one cell and get into the following. The length of this delay depends on the vehicle speed, and it is generated using a random function.

This model was formally specified [7], and, following, we present how to implement the specification using the CD++ tool. We show a part of the specification which includes the top level coupled model and parts of model rB. The top level model includes all the submodels to be executed (presented earlier in figure 6) and their types. Several generators and transducers have been created as instances of the experimental framework (to generate traffic and collect the simulation results). After, rB parameters are included. They follow the previous Cell-DEVS formal specification for coupled models. Then, we include the behavior of the local computing function. A moving car can advance or stay in the cell (if there is a car in the previous cell). The advance is divided in two steps: a car leaving a cell (if the previous is empty), and an empty cell receiving the leaving car. Finally, we show the behavior of the border cell (0,9). Here, the car is let into the crossing using the *send* function (which is only activated if the input value for the cell is 0, meaning there is space in the crossing).

```
[top]
components: rA rB rC rE rF rD1 rD2 rG1 rG2 rH1
rH2 rI1 rI2 railway c1 c2 c3 c4
components: genCar1@generator genCar2@generator
genCar3@generator genCar4@generator
components: cont1@transducer station@Generator
Out : outTrans1 outTrans2
...
[rB]
type : cell      Width : 10      Height : 1
delay : transport     border : nowrapped
neighbors : (0,-1) (0,0) (0,1)
localtransition : segment-rule-1lane
out : y-c-room y-c-car
in  : x-c-car x-c-room
link : x-c-car x-c-car@rB(0,0)
link : x-c-room x-c-room@rB(0,9)
link : y-c-room@rB(0,0) y-c-room
link : y-c-car@rB(0,9) y-c-car
portInTransition : x-c-room@rB(0,9) GoOutCrossing
...

[segment-rule-1lane]
rule : 0 10 { (0,0) = 1 and (0,1) = 0 }
rule : { 1 } 10 { (0,-1) = 1 and (0,0) = 0 }
rule : {(0,0)} 10 { t }

[GoOutCrossing]
rule : { 0 + send(y-c-car,(0,0) ) } 0 {(0,0)=1
and portvalue(ThisPort) = 0 }
rule : {(0,0)} 10 { t }
```

Figure 10. Model definition in CD++

5.2. Crossing definition

This sections shows the translation of the crossing **c1** into Cell-DEVS. The segment definitions provide a the set of inputs and outputs for each crossing. To see this correspondence, the translator checks which segments have **c1** in a border (using the position and traffic direction in each segment). Following this procedure we obtain: $c1_{in}$ = { rG1, rI2, rH2, rD1 } and $c1_{out}$ = { rG2, rI1, rE, rH1, rD2 }. The crossing must include one cell for each lane in a coupled segment. Then, we see the number of lanes in each segment, obtaining:

$$k = \sum_{t \in (C1_{in} \cup C1_{out})} n = 4+4+2+2+2+2+1+2+2 = 21.$$

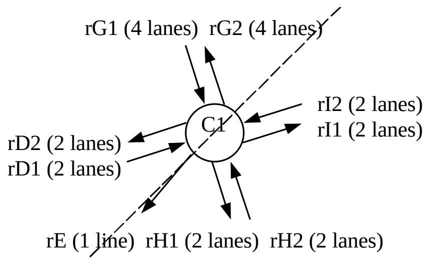

Figure 11 . Choosing input/output segments.

Each segment is coupled with its corresponding cell in the crossing. This correspondence is defined by an ordering given by the incidence angle between the lanes and the line y = 0. The bigger the angle, the higher the position assigned to the segment in the crossing. Therefore, the first cells are coupled with rG2, the following with rG1, etc. This information allows us to define the crossings positions used for input and output: In = {4,5,6,7,10,11,15, 16,19,20}; Out={0,1,2,3,8,9,12,13,14,17,18}. Then, we create the following coupled Cell-DEVS:

$$c1(21, In, Out) = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, \eta, N, C, B, Z >$$
$$= < \{ (0,i) / 0 \le i<21\}, (0,i) / 0 \le i<21\}, I, \{0,1\}, \{0,1\}, 1, t_1=21, 3, \{(0,-1), (0,0), (0,1)\}, C, \{\varnothing\}, Z >$$

with **I** = $<P^x, P^y>$ where $P^x = \{<X_{\eta+1}(0,i), binary> / 0 \le i < 21 \}$ and $P^y = \{<Y_{\eta+1}(0,i), binary> / 0 \le i < 21 \}$.

This crossing is a Cell-DEVS with 21 cells, where the positions of the In set represent inputs to the crossing, and those in Out, outputs. Each output cell is connected to a segment, which will be informed of the present status of a crossing cell. The input cells will receive cars from the segment. The Z function is built using the influence information derived from the translation of the In/Out sets. Therefore, we will consider the following influences: {(0, 0), (0,1), (0,2), (0,3)} with rG2; {(0, 4), (0,5), (0,6), (0,7)} with rG1; {(0, 8), (0,9) } with rD2; {(0, 10), (0,11) } with rD1; {(0, 12) } with rE; {(0, 13), (0,14) } with rH1; {(0, 15), (0,16) } with rH2; {(0, 17), (0,18) } with rI1; {(0, 19), (0,20) } with rI2.

This specification can be written in CD++ as follows:

```
[c1]
type : cell     Width : 21      Height : 1
delay : transport     border : wrapped
neighbors : (0,-1) (0,0) (0,1)

in : x-c-room0 x-c-room1 x-c-room2 x-c-room3 x-c-
car4 x-c-car5 x-c-car6 x-c-car7 x-c-room8
in : x-c-room9 x-c-car10 x-c-car11 x-c-room12 x-
c-room13 x-c-room14 x-c-car15 x-c-car16
in : x-c-room17 x-c-room18 x-c-car19 x-c-car20
link : x-c-room0 x-c-room@c1(0,0)
link : x-c-room1 x-c-room@c1(0,1)
...
link : x-c-car20 x-c-car@c1(0,20)
out : y-c-car0 y-c-car1 y-c-car2 y-c-car3 y-c-
room4 y-c-room5 y-c-room6 y-c-room7 y-c-car8
out : y-c-car9 y-c-room10 y-c-room11 y-c-car12 y-
c-car13 y-c-car14 y-c-room15 y-c-room16
out : y-c-car17 y-c-car18 y-c-room19 y-c-room20
link : y-c-car@c1(0,0) y-c-car0
...
link : y-c-room@c1(0,10) y-c-room10
...
```

Figure 12. Model definition in CD++

Every cell is provided with input/output ports. The input ports of output cells are used to verify if the input segments are busy (using the x-c-room ports). Input cells know the presence of cars in a segment through the ports x-c-car. Output cells use the y-c-car ports to inform to the segments that there is a car (transmitted using the *send* function). Likewise, input cells use the y-c-room output ports to inform the segment that there is space in the crossing for a new car. Finally, links should be created to connect the cells in the crossing with the border cells in the input/output segments.

6. Simulation Results

The constructions in this city section were implemented using CD++. Then, we run several tests providing different experimental frameworks. We will show the results obtained by changing the basic city constructions and experimenting different solutions to given problems.

Every test executed 10 minutes of simulated time. Different vehicles were injected in the model using Poisson generators with different frequencies in different segments (rG1, rH2, rA and rI2, as shown in Figure 6). Transducers gathered global information about the section, collecting the number of cars injected in the sector, and the number of cars getting out of the section.

The following figures show the input/output ratio or the number of cars in the section in fixed periods (1 simulated minute). These measures allows us to analyze the congestion in the area. Two basic scenarios, related with the number of cars injected, were considered: peak-hour and off-peak times. The peak case was implemented by increasing the number of cars injected in segments rG1 and rH2 (those where the traffic increases in the actual city section during rush hours).

The following figure reflects a real problem affecting this section, and the simulated results of a possible solution.

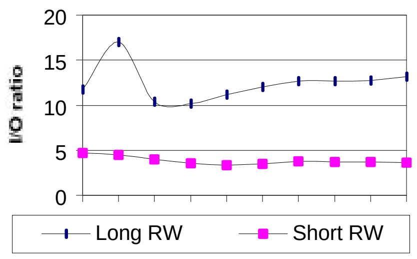

Figure 13. I/O ratio using different delays for the level crossing.

As it can be seen in figure 6, there is a train station close to the crossing c1 (crossing the segment I). At present, the level crossing uses an automatic barrier that closes when a train is approaching to the station. Therefore, while a train is at the station, the barrier is closed. We have simulated this fact using a n average delay of 4 minutes for the level crossing construction. Then, we have changed the delay to 1 minute, supposing we use a controller to detect when the train departs. This change improved the input/output ratio of the sector in average, by a factor of 3, with a maximum of 4.5. The present management of the barrier produces complex traffic jams in the sector that could be easily avoided.

The following experiments compare the congestion degree in the city section during peak times. As we can see, the number of cars in some cases is more than the double of non-peak hours. We also can see that, even we have half of the cars in non-peak hours, the input/output ratio is, at least, of three. This relationship reflects the complex shape of this section, which produces congestion. We also see that the input/output ratio is higher in peak hours.

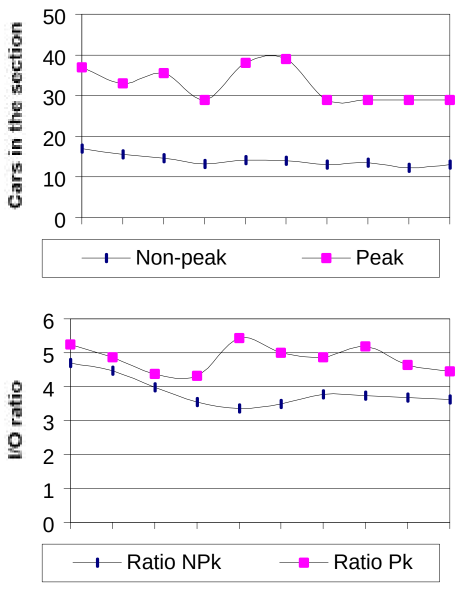

Figure 14. Non-peak traffic against peak traffic (a) Number of cars in the area; (b) I/O ratios.

A final set of tests analyzed the influence of a pothole in the area (cells 4 and 5 of crossing c1). As explained earlier, the behavior of a car crossing a pothole is the same than for other cells, but speed is reduced.

We can see that the input/output ratio was reduced in about 1/3 of the case when the pothole is fixed. The number of cars in the area is the half when the pothole is fixed, as the cars can leave the section earlier, improving traffic flow. The number of cars in the area does not change as abruptly as in the previous cases, because the pothole produces a speed reduction in the main crossing. Therefore, once the cars have crossed the hole, they go out keeping number of cars in the area constant.

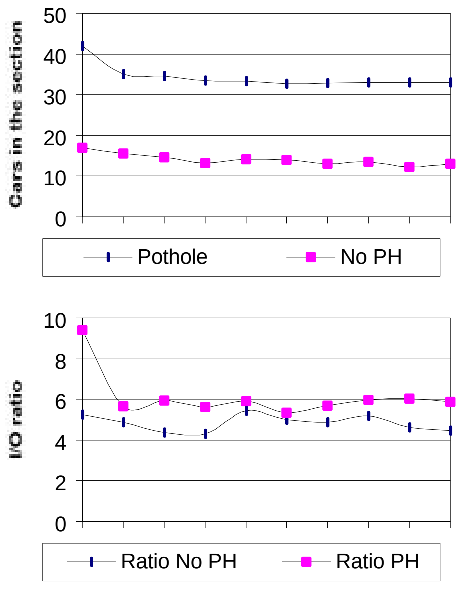

Figure 15. Peak traffic with a pothole(a) Number of cars in the area; (b) I/O ratios.

7. Conclusion

We showed how to use ATLAS to define city sections analyzing traffic conditions. The language allows to define a static view of including different components. This approach provides an application-oriented specification language, which allows the definition of complex traffic behavior using simple rules for a modeler. The models are formally specified, avoiding a high number of errors in the application, thus reducing the problem solving time. The basic idea is to improve the development of the simulations using a simple procedure:

1. A city section is formally specified using ATLAS, according to a shape described by a map. The description can be validated (for instance, verifying positions of crossings, places for parking, trucks flow in forbidden places, etc.).

2. The city section is translated to the tool CD++ (or any other tool allowing the description of DEVS and Cell-DEVS models). The translated models are also specified formally, and its correctness was proved, avoiding errors in their definition.

3. An experimental framework is defined to decide which kind of experiments will be executed over the area. Once these parameters are defined, the simulations can be run.

4. The shape of the section or the traffic flow conditions can be changed easily, allowing to analyze several factors and to provide solutions to existing problems.

This strategy reduces development times because the provision of a solution for a given problem usually implies changes in the model specification and new execution of the experiments, with the goal of making comparisons. This introduces high development costs (moreover in testing) that have been avoided: formal specification mechanisms are used in each phase of the process, using automatic translation procedures.

At present, the specification language has been implemented [8]. In addition, a graphical user interface is being defined, to allow easy definition of the models. The GUI will validate the static model based on information given by the map and the constructions used.

REFERENCES

[1] Davidson, A.; Wainer, G. "Specifying control signals in traffic models". In *Proceedings of AI, Simulation and Planning in High Autonomous Systems, AIS'2000*. Tucson, Arizona. U.S.A. 2000.

[2] Davidson, A.; Wainer, G. "Specifying truck movement in traffic models using Cell-DEVS". In *Proceedings of the 33rd Annual Simulation Symposium*. Washington, D.C. U.S.A. 2000.

[3] Wainer, G. and Giambiasi, N. "Timed Cell-DEVS: modelling and simulation of cell spaces". To appear in *Discrete Event Modeling & Simulation: Enabling Future Technologies*, Springer-Verlag. 2001.

[4] Zeigler, B.; Kim, T.; Praehofer, H. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press. 2000.

[5] Chopard, B.; Queloz, P. A.; Luthi, P. “Cellular Automata Model of Car Traffic in two-dimensional street networks”. J. Phys. A, vol. 29 , pp. 2325-2336, 1996.

[6] Rodríguez, D.; Wainer, G. "New Extensions to the CD++ tool". In *Proceedings of Summer Computer Simulation Conference*. Chicago, U.S.A. 1999.

[7] Davidson, A.; Wainer, G. "ATLAS: a language to specify traffic models using Cell-DEVS". *Technical Report 00-003, Departamento de Computación, FCEN/UBA*. Submitted. 2000.

[8] Lo Tártaro, M.; Torres, C.; Wainer, G. "TSC: traffic simulation compiler for ATLAS" (in Spanish). *Internal report, Departamento de Computación*. FCEN/UBA. 2000.

Keywords: traffic simulation, DEVS models, Cellular models, Cell-DEVS models, Modelling methodologies, Simulation support systems: environments.