VEHICLE ROUTING IN CELL-DEVS MODELS OF URBAN TRAFFIC

Andrea Díaz Verónica Vázquez Gabriel Wainer

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

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

KEYWORDS
Traffic models, DEVS, Cell-DEVS, cellular models.

ABSTRACT

ATLAS is a specification language defined to outline city sections for modelling and simulation of 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 show how to provide routing behavior. Based on routing information, the high level specifications are translated into cellular models that execute those specifications to describe path selections and vehicle routing in the city.

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 (Davidson and Wainer 2000a). 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 behavior is automatically set-up. 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 (Zeigler et al. 2000) and Cell-DEVS models, providing the benefits of a formal approach. Cell-DEVS (Wainer and Giambiasi 2001) was proposed to describe cell spaces as DEVS models with timing delays. 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.

We have extended the static specifications to entitle vehicle routing. The goal is that, providing information about congestion in an area and the conflictive points, a modeler can evaluate structural alternatives to congestion problems. Drivers can be provided with information to avoid bottlenecks. The basic behavior needed to solve this problem include the definition of traffic routing, which will be explained in the following sections.

THE ATLAS MODELLING LANGUAGE

ATLAS (Davidson and Wainer 2000a) is a specification language built on top of DEVS and Cell-DEVS formalisms. A DEVS model can be composed by atomic submodels combined into coupled models. A DEVS atomic model is described as:

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

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: $d_{\mathbf{int}}$ manages internal transitions, $d_{\mathbf{ext}}$ external transitions, l 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.

Cell-DEVS is an extension of DEVS models, especially devoted to define cellular models. Each cell is defined as an atomic DEVS model, and a procedure to couple cells is depicted. Timing delays allow defining different timing behavior. The atomic models (the cells) can be described as:

$$TDC = < I, X, Y, \theta, N, delay, d, \delta int, \delta ext, \tau, \lambda, D >$$

X defines the external inputs, ***Y*** the external outputs, and ***I*** is the interface of the model. q is the cell state definition, and **N** is the set of inputs. **Delay** defines the kind of delay for the cell, and **d** its duration. Finally, there are several functions: $d_{\mathbf{int}}$ for internal transitions, $d_{\mathbf{ext}}$ for external transitions, t for local computations (which use the state values of the neighborhood to compute the future value of a cell), l for outputs and **D** for the state's duration. Each cell takes the set of inputs and computes the cell's future state using the t function. The delay allows deferring the transmission of the results. This behavior is defined by the $d_{\mathbf{int}}$, $d_{\mathbf{ext}}$, l and **D** functions. The modeler only must focus in defining the local computing function, the kind of delay and its length. The remaining parameters are defined by the formalism.

A Cell-DEVS coupled model is defined by:

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

Here, $\mathbf{X}_{list}$ and $\mathbf{Y}_{list}$ are the input/output coupling lists, used by ***I*** to define the interface of the model. ***X*** and ***Y*** represent the input/output events. 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. The cell space is defined by **C**, together with **B,** the set of border cells, and **Z** the translation function. For coupled models, the modeler only has to focus in the neighborhood shape, the size and dimension of the model, the definition of the border set, and the coupling lists.

ATLAS allows representing 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 $\in$ N $\wedge$ a, dir $\in$ {0,1} }, where **p1** and **p2** represent the boundaries of each segment (City = { (x,y) / x, y $\in$ ***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.

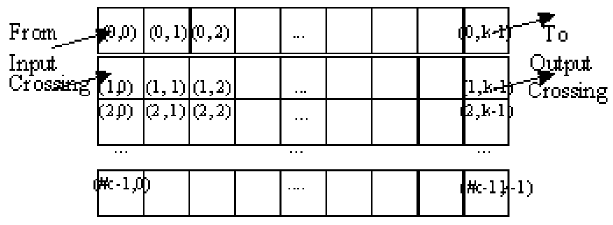

Figure 1: A Segment with More than Four Lanes

In (Davidson and Wainer 2000b)a complete set of rules was defined for streets with one to five (or more) lanes. Different models were included because in each case a different behavior is defined for the borders. The speed of each vehicle is represented by a transport delay. This delay is generated using a random function related with the maximum speed allowed in the street.

- **Crossings**: they are points in the plane where several segments intersect. They are specified as: Crossings = { (c, max) / c $\in$ City $\wedge$ max $\in$ 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. A car in the crossing has higher priority to obtain a position into the ring than the cars out of the crossing.

The Crossings set represents places of the City where the road sections joins. Each crossing can connect any number of road sections, which will be its inputs or outputs. The set of crossings is defined as follows:

Crossings = { c / $\exists$ t ,t' $\in$ RoadSections $\wedge$ 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) }

Every crossing in this set is represented as a ring of cells where the vehicles advance. In any given instant, they can get to another path. A car in the intersection has higher priority to obtain a position into the ring than the cars outside the crossing. The cars advance continuously in order to avoid deadlocks.

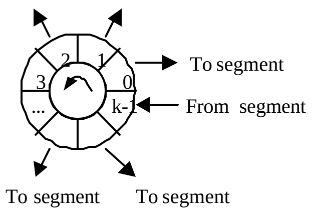

Figure 2: Crossing

Once the components of a city section are defined, complex behavior can be analyzed by using other constructions of ATLAS. Definitions for traffic lights, railways, men at work, street holes, transit signals and parked cars are included. Finally, special behavior has been defined for special vehicles: trucks, vans and high priority cars (ambulances, police, firefighters). The following sections will be devoted to present the behavior for road sections and crossings with different constraints in the traffic flow.

- **Traffic lights**: crossings with traffic lights are defined as: TLCrossings = { c / c $\in$ Crossings }. Here, *c* $\in$ *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 $\in$ RailTrack }, where RailTrack = { (s, δ, seq) / s $\in$ Segments $\wedge$ $\delta \in$ N $\wedge$ seq $\in$ 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 (d). Finally, a sequence number (**seq**) is assigned to each level crossing, defining its position in the *RailTrack*.

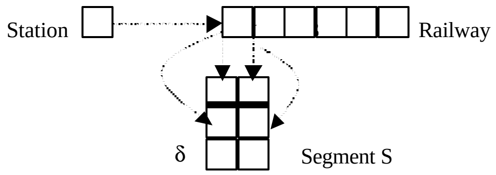

Figure 3: Level Crossing Definition.

- **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 (d), 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.

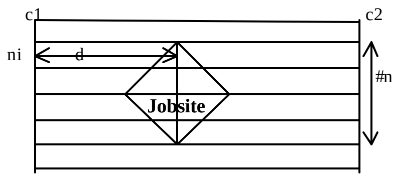

Figure 4: Segment with Men at Work

- **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.

- **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))] }

VEHICLE ROUTING

The original definitions used for ATLAS models were based on random routing. Every time a car arrives to a crossing, the following route is chosen at random. The basic idea is now to include a way to define the routing information for the vehicles.

We have used an approach based in Origin/Destination (OD) matrixes. This tool provides information about routes and transportation between different zones or regions. There are several methods to estimate OD matrixes. Using the definition of a region (represented as a directed graph), the methods use the available information (traffic flow, delays existing in each link of the graph). We will suppose that a OD matrix has been provided, and it will be used by the simulator to make decisions related to vehicle routing.

There are several approaches used to implement OD matrixes, and we have chosen an approach based on a road table. Each register in this table will specify a road connecting a pair of origin/destinations, and the time a car spends in that road. Different tables can be used according to different parameters (for instance, date, time, and type of vehicles). Therefore, the table will have the following structure:

Time Vehicle type
{ ID Origin-node Destination-node {link1 link2 ... linkn} Travel-time }

The structure of the OD matrix and the function used to make the routing decisions can be changed without affecting the simulation models. In this way, both problems can be treated independently. In a first stage, the city shape is defined using ATLAS. Then, a directed graph can be built based on the segment and crossing identifications. Using this graph, and OD matrix can be built. The simulation models devoted to represent routing use a function that queries the OD matrix and provides an answer.

Using this approach, we define a static route for each car, which will not be changing during all the simulation. Using the segment/crossing definition, we can build a graph representing the structure of the city section to be simulated. Using this base, we built an origin/destination matrix. For each pair origin/destination in the matrix, we build a route using the Dijkstra algorithm for shortest paths in a graph. Using this information, we build a complete acyclic route from an input crossing to an output crossing.

Every car is initialized with the route to be used. The original definitions of ATLAS constructions were modified to allow this behavior. The first modification includes an unique identification for each segment/crossing, which will be used with routing purposes. Besides the original port definition (*car* and *room*, indicating that there is a car or space in the origin cell), the coupled models defined for each segment/crossing include now a new port devoted to transfer the routing path (*path* in the following figure).

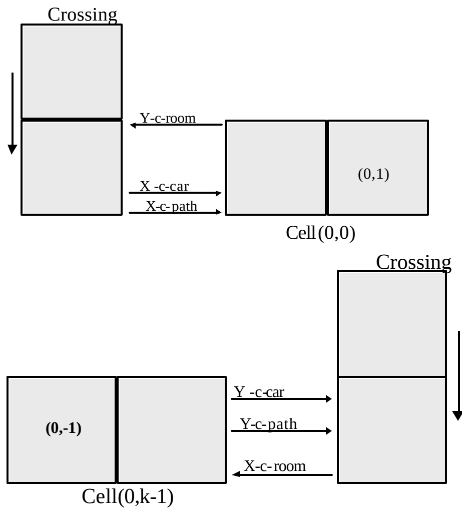

Figure 5: Coupling of the Border Cells

The behavior for the border cells was changed accordingly. This new behavior was also extended for models from 1 to 5 or more lanes (each of them must be defined in a different way due to the definition of the border cells for each of the models).

After changing the segment definition, the crossing were also changed. In this case, the crossings will be in charge of doing the routing definitions using the O/D matrixes. A crossing is defined as a Cell-DEVS coupled model

$$\textbf{\textit{Crossing(k, In, Out, maxc)}} = < \textbf{\textit{Xlist, Ylist, I, X, Y, n, \{t}}}_1,\ldots,t_n\}, h, \textbf{\textit{N, C, B, Z, select}} >$$

where,

Ylist = { (0,i) / $0 \leq i < k$}
Xlist = { (0,i) / $0 \leq i < k$}
I = $<P^x, P^y>$, with P^x = {$<X_{\eta+1}(0,i)$, binary>, $<X_{\eta+2}(0,i)$, binary>, **$<X_{h+3}(0,i)$, Natural>/ 0 £ i < k** }, P^y = {$<Y_{\eta+1}(0,i)$, binary>, $<Y_{\eta+2}(0,i)$, binary>, **$<Y_{h+3}(0,i)$, Natural> / 0 £ i < k** }

Each cell in the crossing is defined by:

$$C_{0j} \textbf{\textit{ (cross-No)}} = < \textbf{\textit{I, X, S, Y, N}}, d_{int}, d_{ext}, \textbf{\textit{delay, d}}, t, l, \textbf{\textit{D}} >$$

I = $< \eta, P^x, P^y >$, $\eta = 3$, P^x = { (X_1, Record), (X_2, Record), (X_3, Record) }, P^y = { (Y_1, Record), (Y_2, Record), (Y_3, Record) }; ***X, Y*** $\in$ N,

S* = *(Car, crossing, path, segment-no), where

$$Car = \begin{cases} 1 \text{ there is a vehicle} \\ 0 \text{ otherwise.} \end{cases}$$

crossing $\in$ N: unique identifier of the crossing;

$$path = \begin{cases} \{t_1.t_2\ldots\ldots t_n\} \text{ where } t_i \in N \wedge (\forall i (\exists r \in \\ \qquad \text{Segments/ segment-no}(r) = t_i)) \\ 0 \text{ otherwise.} \end{cases}$$

Segment-no $\in$ N: identifier of the segment to which the cell is connected.

N = { (0,-1), (0,0), (0,1) }
delay = transport
d = (speed(maxc))
t is in charge of defining the car behavior in the cell, according to the new routing scheme. Different behavior is defined for the input and output cells, using the O/D matrixes.

We now show the definition of a simple example using the new definitions. The example was implemented using the CD++ tool (Rodríguez and Wainer 1999). The model consists of 5 segments and 4 crossings, and will describe the behavior of traffic using the routing scheme defined previously. The implementation is a simplification of the theoretical definition. We have used different state variables in each cell: the car existence, the route covered up to the moment and the state of the segments. The path is specified by a real number with the following format: d.ddddd, where 1=< d <=9 is the identifier of a segment.

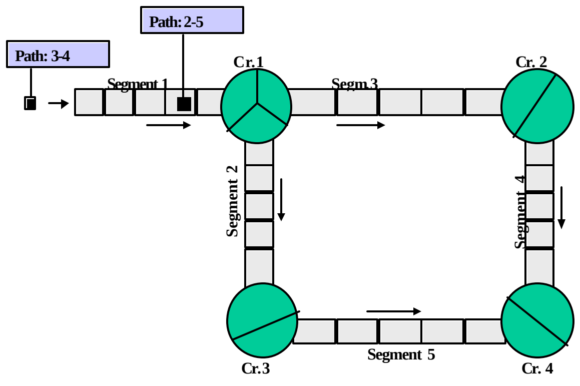

Figure 6: A City Section

The following figure shows the simulation result in the segment 1. The first row represents the state variable showing the presence of a car. The second line represents the path to be followed by the car. The first car will take the path 2-5, whereas the second will go through the path 3-4. This behavior can be found in figure 7.

```
        Time: 00:00:00:000
          0     1    2      3      4
     +------------------------------+
   0 |                    1.00       |
   1 |                    2.50       |
     +------------------------------+

        Time: 00:00:00:010
  26 109   5 109
```