Defining Congestion Control Mechanisms in ATLAS

Gabriel Wainer

Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON. K1S 5B6. Canada.
gwainer@sce.carleton.ca

Andrea Díaz **Verónica Vázquez**

Departamento de Computación
Universidad de Buenos Aires
Pabellón I. Ciudad Universitaria (1428)
Buenos Aires. Argentina.
{adiaz, vvazquez}@dc.uba.ar

Keywords: traffic models, DEVS, Cell-DEVS, cellular models.

Abstract

The ATLAS specification language entitles the definition of city section models used to simulate traffic flow. The model static behavior is characterized by parametric definition. Then, language constructions are translated into DEVS and Cell-DEVS models to represent the system dynamic behavior. We have extended the original rules defined for the language constructions to include routing mechanisms and congestion avoiding techniques. An example of the application of the proposed strategies illustrates these concepts.

INTRODUCTION

Traffic analysis and control complexity is increasing in a daily basis, and simulation based studies have proven to be helpful to improve traffic control, avoid pollution, traffic jams, etc.

ATLAS (Advanced Traffic LAnguage Specifications) is a high level specification language defined to represent city sections as cell spaces [1]. It is focused to analyze detailed behavior of traffic (microsimulations). 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 components. Therefore, a modeler can concentrate in the transit problems to be solved.

The constructions defined in this language are used to define a static view of the system to be modeled. System dynamic behavior has been defined by creating a mapping into DEVS [2] and Cell-DEVS models [3]. Cell-DEVS was proposed to describe cell spaces as DEVS models with explicit delays. Using Cell-DEVS, a cellular model can be described as a discrete event model, and transport and inertial delays allow accurate timing description.

This work was partially supported by ANPCYT research project 11-04460 (Argentina) and NSERC (Canada).

The article focuses on the definition of congestion monitoring techniques. Congestion information about conflictive points can be useful to evaluate structural alternatives to congestion problems, or giving the drivers information to avoid traffic jams. The language constructions were extended, allowing dynamic traffic routing.

We briefly recall the DEVS and Cell-DEVS formalisms, and explain the main components of the specification language. Then, the congestion monitoring techniques employed are presented. finally, the results of an implementation example are introduced.

DEVS AND CELL-DEVS FORMALISMS

ATLAS is a specification language built on top of DEVS and Cell-DEVS formalisms. DEVS formalism permits to specify discrete events systems using a modular description. A model is seen as composed by atomic submodels than can be combined into coupled models. DEVS is a discrete event paradigm. It uses a continuous time base, which allows accurate timing representation. Precision of the conceptual models can be improved, and CPU time requirements reduced. A DEVS model is seen as composed of atomic submodels than can be hierarchically combined into coupled models. A DEVS atomic model is described as:

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

X is a set of input events, **S** defines the state variables, and ***Y*** is a set of output events. The function d_{int} manages internal transitions, d_{ext} external transitions, l the outputs, and **ta** the elapsed time. A DEVS coupled model is defined as:

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

Here, $\boldsymbol{X}$ is the set of input events, and $\boldsymbol{Y}$ 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 $M_i = < X_i, S_i, Y_i, \delta_{int_i}, \delta_{ext_i}, ta_i >$. $\mathbf{I_i}$ 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, especially devoted to define cell spaces. Each cell is defined as an atomic DEVS, and a procedure to couple cells is depicted. Timing delay constructions let the modeler to define the cell timing behavior. Each cell, built as an atomic model, can be described as:

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

$\boldsymbol{X}$ defines the external inputs, $\boldsymbol{Y}$ the external outputs. 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, l for outputs and **ta** for the state's duration. Each cell uses the set of inputs to compute the cell's next state using the t function. The delay allows to defer the transmission of the results. This behavior is defined by the $d_{\mathbf{int}}$, $d_{\mathbf{ext}}$, l and **ta** functions. A modeler only focuses in defining the local computing function, the kind of delay and its length.

A Cell-DEVS coupled model is defined by:

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

Here, $\mathbf{X_{list}}$ and $\mathbf{Y_{list}}$ are the input/output coupling lists. $\boldsymbol{X}$ and $\boldsymbol{Y}$ 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**, **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.

In ATLAS, the structure of a city section is represented by a set of streets connected by crossings. Some of the language components include:

. **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, **n** is the number of lanes, and **dir** represents the vehicle direction. The **a** parameter defines the shape of the segment, and **max** is the maximum speed allowed.

. **Crossings**: they represent the places where the streets (represented as sets of segments) are gathered. Each crossing can connect any number of segments. They can be defined as: Crossings = $\{ c / \exists t, t' \in Segments \wedge t = (p1, p2, n, a, dir, max) \wedge t' = (p1', p2', n', a', dir', max') \wedge t \neq t' \wedge (p1 = c \vee p2 = c) \wedge (p1' = c \vee p2' = c) \}$

. **Traffic lights**: crossings with traffic lights are defined as: TLCrossings = $\{ c / c \in Crossings \}$. Every $c \in TLCrossings$ is a set of 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 = $\{ (s, \delta, seq) / s \in Segments \wedge \delta \in N \wedge seq \in N \}$. *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*.

. **Men at work**: they are specified as: Jobsite = $\{ (s, ni, \delta, \#n) / s \in Segments \wedge s = (c1, c2, n, a, dir, max) \wedge ni \in [0, n-1] \wedge \delta \in N \wedge \#n \in [1, n+1-ni] \wedge \#n \equiv 1 \bmod 2 \}$. Here, each $(s, ni, d, \#n) \in Jobsite$ is related to 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 where the cars cannot advance.

. **Traffic signs**: they are defined by: Control = $\{ (s, t, \delta) / s \in Segments \wedge \delta \in N \wedge t \in \{$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.

CONGESTION MONITORING

Traffic flow rate in a city section can influence the vehicle movement and the decisions taken by the drivers. Most existing modeling approaches based on cell spaces do not consider the information related to congestion, or provides mechanisms to reproduce vehicle routing. Our goal in this section is to present an extension to ATLAS to represent this behavior.

To entitle the definition of vehicle routing, path information for the cars should be included. We have used an approach based in Origin/Destination (O/D) matrixes. These arrays provide data about routes and traffic flow between different regions in a city. Each element in the matrix represents the amount of traffic or the delays between the origin and the destination. O/D matrixes can be built using the definition of a region represented as a directed graph. Then, the available information (traffic flow, delays existing in each link of the graph) is inserted in the matrix. We suppose that a O/D matrix is provided for the region to be modeled, and we will be use it to make decisions related to vehicle routing.

There are several ways to implement O/D matrixes, and we have chosen an approach based on a road table. Each register in this table specifies a road connecting a pair of origin/destinations, and the time a vehicle spends in that road. The table has the following structure:

Time Vehicle type
{ ID Origin Destination {link1...linkN} Travel-time }

Here, *Time* and *Vehicle* types are used to build different tables according to different parameters. The structure of the O/D 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.

A traffic model is built by defining the city shape using ATLAS. Then, a directed graph can be built based on the segment and crossing identifications. Using this graph, the O/D matrix can be created. The simulation models devoted to represent routing use a function that queries the O/D matrix and provides a route to be followed by a vehicle.

Once these basic static routing strategies were defined, the language was extended by adding a measure of traffic congestion. Based on this information and using the O/D matrixes, the cars can change their original routes. A new DEVS model, devoted to monitor congestion, was added. Now, every segment is provided with a controller to measure the number of cars. This model is defined as:

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

X = {< x-r-carIn, N>, < x-r-carOut, N>}
Y = {< y-r-weight, N>}
S = k ∈ N representing the number of cars in the segment under consideration,

```
δext () {
        when (x-r-carIn = 1)
                k = k + 1;

        when (x-r-carOut = 1)
                k = k – 1;
passivate;
}
λ () { send k through the port y-r-weight }
```

Once the DEVS congestion controllers were defined, the border cells of the Cell-DEVS representing the segments were changed to transmit information about the cars arriving or leaving a segment. New input/output ports were added to transmit this data to the coupled models corresponding to the crossings, depicted in the following figure:

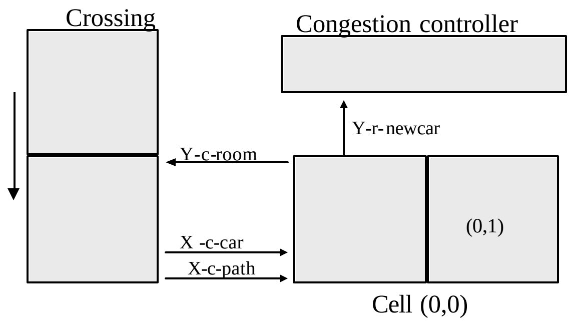

Figure 1. Coupling scheme of congestion controllers, crossings and segments.

The behavior of these models is similar to the ones presented in [1]. The rules used in the border cells have been redefined to include the congestion management definition. Now, these cells must inform the number of cars entering and leaving the segment to the DEVS congestion monitor. For instance, for one-lane segments, this specification is now translated into a Cell-DEVS defined by:

$C_0(segment\_no) = < I, X, S,Y,N,d_{int},d_{ext},delay,d,t, l,D >$

I = <η, P^X, P^Y>, with $\eta = 3$, P^X = { <X_1, Record>, <X_2, Record>, <X_3, Record> } and P^Y = { <Y_1, Record>, <Y_2, Record>, <Y_3, Record> }
X, $Y \in$ N
S : {s, phase, σqueue, σ}, with s = (destination, path)

$$destination \in N = \begin{cases} \neq 0 \text{ if there is a vehicle in the cell} \\ 0 \text{ otherwise.} \end{cases}$$

$$path = \begin{cases} \{t_{1...}t_n\}\ ti \in N \wedge (\forall i(\exists\ r \in \text{Segments}/ \ \text{Segment\_no}(r) = ti\ )\ ) \\ 0 \text{ otherwise.} \end{cases}$$

N = { (0,-1), (0,0), (0,1) }
delay = transport
d = speed(max)
λ, δ_{int} y δ_{ext} are defined by Cell-DEVS with transport delays.
τ: S x N $\rightarrow$ S. The behavior of the local computing function can be roughly defined by:

New state	Neighborhood

Dest = Dest(0,-1)
Path = Path(0,-1)
	Dest(0,-1) != 0 and Dest(0,0) = 0

Dest = 0
Path = 0
	Dest(0,0) != 0 and Dest(0,1) = 0

Figure 2. Local computing function for the segments.

In this case, the first rule represents a vehicle arriving to the cell, coming from the previous cell. The second rule represents the advance of the vehicle to the following cell. As we can see, the identifier of the destination cell represents the vehicle, and the path of the vehicle is transferred between cells.

After defining the behavior for the controller and the segments (with 1 to 5 lanes), a new coupled model for the crossings was created.

Then, the rules used for defining the crossing behavior were modified. Now, every crossing will receive information from the congestion controller, and, based on the availability of paths to arrive to the same destination and the congestion information, a routing decision is taken. The models now include the definition of routing mechanisms associated with each cell, defined as:

$C_j(crossing\_no) = < I,X,S,Y,N,d_{int},d_{ext},delay,d,t,\ l,\ ta >$

I = < η, P^X, P^Y>, with η = 3, P^X = { <x-t-destinat, Record>, <x-c-congestion, Record>, <x-t-path, Record> }, P^Y = { <y-t-room, Record>, <y-c-vehicle, Record>, <y-c-path, Record> }.
X, *Y* $\in$ N;
S: {s, phase, σqueue, σ}, with s = (destination, path, crossing_no, segment_no), with

$$\text{destination} = \begin{cases} \text{!=0 if there is a vehicle (representing the destination crossing).} \\ 0 \text{ otherwise.} \end{cases}$$

$$\text{path} = \begin{cases} \{t_1.t_{2...}t_n\},\ ti \in N \wedge (\forall i(\exists\ r \in \text{Segments} \ / \text{segment\_no}(r) = ti\ )\ ) \\ 0 \text{ otherwise.} \end{cases}$$

crossing_no $\in$ N: crossing identifier;
segment_no $\in$ N: identifier of the segment to which the output cell of the crossing is connected.
N = { (0,-1), (0,0), (0,1) }
delay = transport
d = speed(maxc)
λ, δ_{int} y δ_{ext} are defined by Cell-DEVS with transport delays.
τ: S x N $\rightarrow$ S. The behavior of the local computing function can be roughly defined by:

New state	Neighborhood

Dest= Dest(0,-1)
Path= Path(0,-1)
Crs_no= crs_no(0,0)
Seg_no= seg_no(0,0)
	Dest(0,0)= 0 and Dest(0,-1) != 0
	Send(1, y-t-room)

Dest= (x-t-dest)
Path= (x-t-Path)
Crs_no= crs_no(0,0)
Seg_no= Seg_no(0,0)
	Dest(0,0)=0 and Dest(0,-1)= 0 and (x-t-dest) != 0 and (x-t-dest)!= crs_no(0,0) and !Congestion((x-c-congest, next-seg(path(0,-1)), seg_no (0,0)))
	Send(1, y-t-room)

Dest= (x-t-dest)
Path= New_Path(Crs_no(0,0), Dest(0,-1), Path(0,-1))
Crs_no= crs_no(0,0)
Seg_no= Seg_no(0,0)
	Dest(0,0)= 0 and Dest(0,-1)= 0 and (x-t-dest) != 0 and (x-t-dest) != crs_no(0,0) and Congestion((x-c-congest, next-segment(path(0,-1), seg_no(0,0)))
	Send(1, y-t-room)

Dest= 0
Path= 0
Crs_no= crs_no(0,0)
Seg_no= Seg_no(0,0)
	Dest(0,0)= 0 and Dest(0,-1)= 0 and (x-t-dest) != 0 and (x-t-dest)= crs_no(0,0)
	Send(0, y-t-room)

Figure 3. Rule definition for the crossings.

In this case, the first rule introduced represents the arrival of a vehicle to a cell that was in the crossing and preserves the original path. The second rule represents the input of a vehicle to the crossing that has not arrived to the destination, conserving the original route. The following rule represents the input of a vehicle that changes the path due to congestion in the area. The fourth rule eliminates a vehicle that has arrived to the destination.

The crossing number uniquely defines the crossing where this cell is defined. In this way, an O/D matrix can be used, and the crossing identified to permit different paths to be taken. The cell state represents the existence of a vehicle, the path to be followed, and the crossing identifier. Using this information, every vehicle arriving to the crossing can be routed according to the congestion information.

When a vehicle arrives to a crossing, it will be sent to the input cell, which will be in charge of deciding the vehicle routing. The input cells will use the congestion information sent by the congestion monitors.

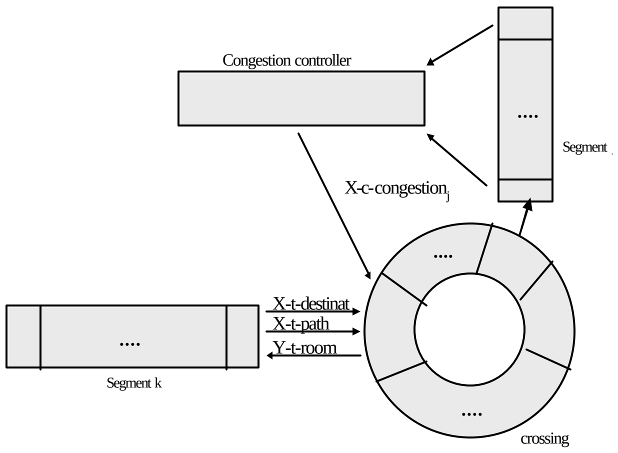

Figure 4. Coupling scheme of output cells of segments, crossings, and congestion controllers.

SIMULATION RESULTS

Once the whole set of new rules and coupled model definitions were defined (see details in [4]), we defined the new set of constructions using the CD++ tool [5]. In this section, we will show the definition of a simple example implementing the new routing techniques.

The following figure depicts a section of the city in which the new routing strategies were tested. The circles represent crossings, and the lines, the segments between two crossings. In this case, we have used one way streets with one lane each.

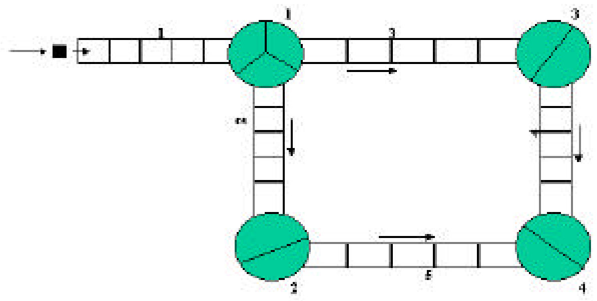

Figure 5. Representation of a city section.

This city section is composed of 5 segments and 4 crossings interconnected. Different cars arrive through the segment 1, and all of the cars will follow the same path (segment2-segment5). The segment 2 includes a pothole delaying the advance of cars. The defin ition of the congestion function considered that a segment with 2 or more cars is congested. The goal of this example is to show how the dynamic routing is achieved.

The segments were implemented as Cell-DEVS models using two state variables. The first variable defines the vehicle destination (or 0 if the cell is empty), and the second state variable include a definition of the path to be followed by the vehicle. The following figure shows the execution results for the segment 1:

```
Time: 00:00:00:010                    Time: 00:00:00:040
      0    1    2    3    4                 0    1    2    3    4
  +-----------------------------+       +-----------------------------+
 0 |4                           |      0 | 4               4           |
 1 |2,5                         |      1 | 2,5             2,5         |
  +-----------------------------+       +-----------------------------+
Time: 00:00:00:050                    Time: 00:00:00:060
      0    1    2    3    4                 0    1    2    3    4
  +-----------------------------+       +-----------------------------+
 0 |    4                 4     |      0 |         4                   |
 1 |    2,5               2,5   |      1 |         2,5                 |
  +-----------------------------+       +-----------------------------+
Time: 00:00:00:110                    Time: 00:00:00:140
      0    1    2    3    4                 0    1    2    3    4
  +-----------------------------+       +-----------------------------+
 0 |    4                 4     |      0 |                        4    |
 1 |    2,5               2,5   |      1 |                        2,5  |
  +-----------------------------+       +-----------------------------+
```

Figure 6. Execution results in Segment 1.

In this example we see that in the simulated time 00:00:010, 00:00:040, 00:00:070 and 00:00:100 new cars have arrived to the cell (0,0). Every vehicle will follow the route segment 2-segment 5. When the cars finish traversing the segment, they will be routed to the crossing 1.

```
 Time: 00:00:00: 000                  Time: 00:00:00:050
          0        1        2                  0        1        2
  +---------------------------+         +---------------------------+
 0 |                          |        0 | 4                         |
 1 |                          |        1 | 2,5                       |
 2 |        2        3        |        2 |        2        3         |
  +---------------------------+         +---------------------------+
Time: 00:00:00:060                    Time: 00:00:00:080
          0        1        2                  0        1        2
  +---------------------------+         +---------------------------+
 0 |        4                 |        0 | 4                         |
 1 |     2,5                  |        1 | 2,5                       |
 2 |        2        3        |        2 |        2        3         |
  +---------------------------+         +---------------------------+
Time: 00:00:00:110                    Time: 00:00:00:120
          0        1        2                  0        1        2
  +---------------------------+         +---------------------------+
 0 |        4                 |        0 | 4                         |
 1 |     2,5                  |        1 | 3,4                       |
 2 |        2        3        |        2 |        2        3         |
  +---------------------------+         +---------------------------+
Time: 00:00:00:130                     Time: 00:00:00:140
          0        1        2                  0        1        2
  +---------------------------+         +---------------------------+
 0 |        4                 |        0 | 4               4         |
 1 |        3,4               |        1 | 2,5             3,4       |
 2 |        2        3        |        2 |        2        3         |
  +---------------------------+         +---------------------------+
```

Figure 7. Execution results in Crossing 1.

This crossing implements the dynamic routing techniques previously explained. The cell 0 of the crossing is an input cell, while the remaining two are used for outputs. The following figure shows the values of the different state variables used in each cell of the crossing. The state variables showed in the lines 0 and 1 represent the vehicle information used for the segments (destination and route). Line 2 represents the segments to which the output cells are connected.

The first two cars arriving to the crossing (in 00:00:050 and 00:00:080) keep the original path (segment 2-segment 5). As the cell 1 in the crossing is connected to the segment 2, the vehicle is sent to the crossing through this cell.

The last two cars arriving to the crossing (at 00:00:110 and 00:00:140) must take a new path, because the congestion function for the segment 2 returns a value representing that the segment is congested. Therefore, the cars ask to the O/D matrix for a new path, and the model returns the path segment 3-segment 4. Then, they leave the crossing through the cell 2, connected to the segment 3, following the rules defined in the previous sections.

CONCLUSION

ATLAS is an application oriented specification language that allows the definition of complex traffic behavior using simple rules. The models are formally specified, avoiding a high number of errors in the developed application, and the problem solving time is highly reduced, allowing analyzing complex behavior in the traffic, and providing new solutions.

In this case, we have extended the original definitions to include complex routing behavior not available in other microsimulation tools. Due to the hierarchical and modular characteristics of DEVS and Cell-DEVS, the inclusion of this new complex behavior was straightforward. The implementation of these techniques in existing DEVS tools allowed us to prove the implementation feasibility of the approach, entitling the future inclusion of the routing mechanisms in the TSC compiler for the ATLAS language [6].

REFERENCES

[1] Davidson, A. and Wainer, G. 1999. "A specification language for traffic modeling and simulation". Technical Report 99-012, Departamento de Computación, FCEN/UBA. Submitted.

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

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

[4] Díaz, A.; Vázquez, V. 2000. "Routing model definition in ATLAS" (in Spanish). M. Sc. thesis. Departamento de Computación, FCEN/UBA.

[5] Rodriguez, D.; Wainer, G. 1999. "New Extensions to the CD++ tool." In *Proceedings of SCS Summer Computer Simulation Conference*, Chicago, USA.

[6] Lo Tartaro, M.; Torres, C.; Wainer, G. "TSC: a compiler for the ATLAS language". 2001. Technical Report, 01-002. Computer Science Dept. Submitted.

Gabriel Wainer received his M. Sc. (1993) and Ph.D. degree (1998) from the Universidad de Buenos Aires, Argentina, and Université d'Aix-Marseille III, France. He is currently Assistant Professor at the SCE Dept. of Carleton University (Ottawa, Canada). He was Assistant Professor at the Computer Sciences Dept. of the Universidad de Buenos Aires, Argentina, being a Visiting Research Scholar at the University of Arizona, Tucson, AZ. He has been the PI of several research projects (ANPCYT - Argentina, Usenix Foundation - USA, NSERC - Canada, etc.). He is author of a book on real-time systems and another on Discrete-Event simulation. He is a member of the Board of Directors of the Society for Computer Simulation International, and a member of a group on standardization of DEVS modeling tools.

Andrea Díaz and **Veronica Vázquez** received their B.Sc. M. Sc. degree at the Universidad de Buenos Aires in 1997 and 2000. They are in high tech firms in Buenos Aires, Argentina.