

Timed Cellular Automata-Based Tool for the Analysis of Urban Road Traffic Models

Camelia Avram, Adina Astilean and Eduardo Valente

Abstract The optimization process of urban transportation in smart cities is strongly connected to the elaboration of specific, efficient models. In this context, this chapter describes a versatile modelling formalism based on timed automata and implemented in the UPPAAL environment for different complex and easily changeable road traffic simulations. Microscopic models based on cellular automata are analysed in order to simulate the behaviour of different vehicles in a specific group of urban streets. The proposed models integrate the main traffic elements present in urban traffic: streets with multiple traffic lanes; different types of vehicles, including automobiles, buses and trams; intersections controlled by traffic lights; bus and tram stops inside and outside of the traffic lane, pedestrian crosswalks; and parallel street parking. The basic concepts are detailed starting from scenarios which first merit to highlight possible modelling techniques and structures and to facilitate a comparative analysis of the limitations of the presented models. The models based on traffic cellular automata (TCA) have appropriate results inside of the urban traffic theory.

Keywords Urban traffic · Modelling · Simulation · Cellular automata
Formal verification

Urban Traffic Theory

Historically, traffic congestion has been regarded as a problem confined to major metropolitan areas. Over the years, the traffic problems that existed in densely developed urban areas have begun creeping into the suburbs. Automobile technology advances have allowed more people to drive and the feverish modern

C. Avram (✉) · A. Astilean
Technical University of Cluj-Napoca, Cluj-Napoca, Romania
e-mail: camelia.avram@aut.utcluj.ro

E. Valente
University of Minho, Guimarães, Portugal

lifestyle has caused traffic congestion problems to appear even in small towns. This hectic scenario has given rise to the need for new solutions that improve traffic circulation, and traffic simulations have played a key role [1].

A good understanding of road traffic dynamics is fundamental to assist the choice between the strategies that are more efficient and more appropriate. In this context, simulations that reproduce the eventual effect of traffic change parameters may be extremely important for an improvement of road traffic circulation [2].

In this chapter, the fundamental concepts of the urban traffic theory are presented. Firstly, the main physical variables involved in the traffic road problems (flow, average speed, and density) are shown, as well as different approaches to express those variables. Afterward, relational diagrams between these variables are depicted and their theoretical behaviours are exposed. A presentation of some microscopic models for traffic road simulation closes this chapter [2].

Fundamental Concepts of the Urban Traffic Flow (Traffic Parameters)

The traffic behaviour can be evaluated by the following variables: flow (J), average speed (v) and density (ρ). These parameters are called traffic parameters, i.e., those variables that help to determine the road condition at a particular time. Traffic flow is defined as the quantity of vehicles that pass through a road section in a given period of time and its units are vehicles per time unit. The average speed is given by space units travelled by those vehicles per time unit. The density is determined by the number of vehicles per space unit [3, 4].

In this section different methods to calculate the urban traffic parameters are presented. The different expressions provide the necessary background for the utilization of urban traffic parameters in the course of this work [2].

The variables' behaviour is defined in the space-time diagrams, which represent each vehicle's trajectory in a period.

The traffic characteristics range in time and space. In order to simplify these variations, commonly average values for the traffic parameters are adopted. These average values may be temporal or spatial values. Thus, there are different expressions for the traffic variables: When one road is considered in one time interval, it is called temporal average; or a lane stretch in a time instant is called a spatial average [2, 5].

Spatial Average

The density for a lane stretch (L) in a time instant (dt) is the number of trajectories (vehicles) on this stretch at the time instant t_1 divided by the considered length (L) [2].

Equation 1 shows the mathematical expression of traffic density.

$$\rho = \frac{n}{L}, \quad (1)$$

where n , is de number of vehicles present in the traffic stretch L .

The vehicles' average speed on this traffic lane stretch can be expressed by the following equation:

$$\bar{v} = \frac{\sum_{i=1}^n v_i}{n}, \quad (2)$$

where v_i is the instant speed of the vehicle i in the considered stretch. For a permanent regime of speed, it can be calculated by:

$$v = \frac{J}{\rho}, \quad (3)$$

In other words:

$$J = \rho \times v, \quad (4)$$

As in this case is considered the vehicles speed on a section (v):

$$J = \rho \times \bar{v}, \quad (5)$$

The traffic flow equation can be written:

$$J = \frac{\sum_{i=1}^n v_i}{L}, \quad (6)$$

Temporal Average

In the previous case, the information is a function of a single time instant from the considered traffic lane stretch. Normally, when is necessary to obtain real data, motion detectors are installed in any traffic lane [2]. Therefore, it is necessary to

define expressions that consider multiple measurements on the same traffic lane section x_k . The traffic flow is given by the number of vehicles that cross a determined lane section (m), in a certain time interval (T), i.e.

$$J = \frac{m}{T}, \quad (7)$$

The vehicle's average speed that crosses the lane section is expressed by:

$$\bar{v} = \frac{\sum_{i=1}^n v_i}{m}, \quad (8)$$

v_j , is the vehicle's speed that crosses this section. Substituting Eqs. 7 and 8 into Eq. 5, the average density of vehicles on that lane section is given by [2, 6]:

$$\rho = \frac{m^2}{T \times \sum_{i=1}^n v_i}, \quad (9)$$

Traffic Flow and Density

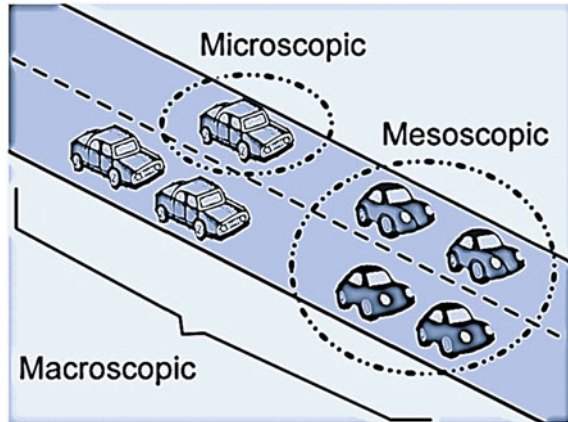
The traffic flow has three different phases [7, 8]:

- Low-density region, called free traffic flow. This phase, generically, allows the driver to achieve a desired speed, approaching the maximum speed permitted.
- Median density region, $c_1 < \rho < c_2$, where the traffic flow is not exclusively defined by density. The traffic configuration influences directly its flow, and may cause a free or congested flow. This region (the middle density) is also known as meta-stable region;
- High-density region, $\rho > c_2$, where the traffic flow drops as the density increases. The large concentration of vehicles causes them to cluster in traffic jams and a vehicle that leaves a place from the traffic jam will find congestion just ahead. This type of traffic is characterized by the behaviour of starting and stopping.

Models for Simulation of Vehicular Traffic

Mathematical models for traffic simulation can be divided into three different approaches: macroscopic, mesoscopic, and microscopic models [9, 5]. Figure 1 represents schematically these three different methodologies (Image adapted from [4]).

Fig. 1 Main strategies for traffic simulation



Macroscopic Models

The macroscopic analysis describes the global behaviour of traffic streams. Therefore, they relate density, flow and average speed parameters of the vehicles. To study its behaviour, the macroscopic approaches apply hydrodynamic rules. This is the reason why this approach is also known as traffic hydrodynamic analogy. For its features and considerations, macroscopic models are successfully applied in the study of high-density traffic, but they cannot provide accurate results easily for rarefied traffic situations [4, 10–13].

Mesoscopic Models

The mesoscopic models represent the behaviour of a group of vehicles, i.e., bases its traffic analysis in a group of vehicles that behaves according to some logical grouping criteria: an expedition, congestion behaviour, etc. [4, 10–14].

Microscopic Models

Microscopic models are the ones that focus the individual behaviour of each vehicle to obtain the global behaviour of a traffic road. They consider the interrelated parameters that determine the vehicle's dynamics. For example, knowing the acceleration of each vehicle at each time instant its position and speed can be known, after a time interval [4, 10–14].

Microscopic Models

A type of microscopic modelling that has been largely used in traffic simulation is based on cellular automata, due to its versatility and simplicity. The first probabilistic model that reproduced the basic traffic conditions with the use of simple transition rules was proposed in [15–18].

One of the most studied models in traffic road simulation are the microscopic models, using the persecution model (car-following), which was developed at the end of 1950s. This model has as first objective the translation of speed variation of the tracker vehicle. The speed variation is a response function of the speed stimulus between a vehicle and the vehicle in front of it, called leader vehicle [2].

Urban Road Traffic Modelling Issues

The formalism adopted for modelling the systems considered in this work (road traffic systems) is timed automata (TA) [12, 13, 19–22, 17, 18]. This formalism is widely used on several domains for modelling behaviour of physical systems due to its non-determinism characteristic [16, 23, 24, 25, 9].

In addition, because the UPPAAL simulation and verification environment is used for performing the simulation and formal verification analysis techniques, the edition of those models was implemented on the editor of this software tool, as performed in [26–30].

Each example presented in this chapter defines a concrete problem and the model's evolution until the creation of the final model that contains all the features needed to implement in the case study.

The main aim of this work is to create a systematic approach for traffic road models. This systematic approach can only be obtained using modular structures. This model also must be easily extendable, reused and contain a large level of detail, in order to be possible implementing the complex scenario proposed. The solution implemented to create such versatility features introduced matrices that provide a compact environment for discretization and modelling the physical environment.

The models presented in this chapter are divided into two levels of complexity. The models of low complexity consist of the following:

- A model with one traffic lane and one automobile.
- Two models are presented to describe a traffic lane with several automobiles travelling simultaneously.
- The complexity increases model by model. The models with a high complexity represent the following situations:
- Four traffic lanes connected by an intersection and the automobiles chooses the next street to continue its journey.

At the end of each subchapter, we present the features, advantages and/or limitations of each proposed model.

Low Complexity Modes

The first models created contain the simplistic interactions between a street and vehicles. These two models were the first approach and the level of detail and degree of realism are not the desired for this work. They were useful for understanding which modelling techniques and structures that have potential to be further developed in order to reach the versatility required.

Model with One Traffic Lane and One Automobile

The first model created used one automobile moving in a street freely, without interactions with other automobiles. The absence of interactions with other automobiles is because only an automobile can travelling inside of the street.

At the beginning of the street and also at the end of a street, motion sensors detect the presence of a vehicle approaching from the beginning of the street (Si1) or leaving the end of the street (So1). The initial conditions of this problem also consider that an automobile enters in the street with velocity different from zero.

In Fig. 2 the physical environment of the first model is schematically represented.

To implement this scenario in UPPAAL, it was necessary develop two automata: the automaton street and the automaton automobile.

The automaton street has five places:

- OUT_of_the_STREET_START: When an automobile is travelling outside of the beginning of the street.
- STREET_START: When the sensor Si1 has detected an automobile at the beginning of the street.
- OCCUPIED: When an automobile is on the street.

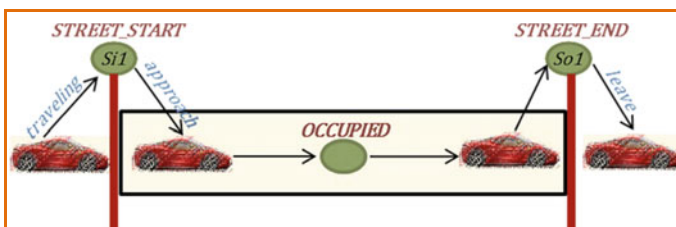


Fig. 2 A single traffic lane with one travelling automobile

- **STREET_END**: When an automobile was detected by the sensor So2 at the end of the street
- **OUT_of_the_STREET_END**: When an automobile is travelling outside of the end of the street.

These are the five states that a street can have when interacting with an automobile. To coordinate the interactions between the street and the automobile, it was necessary to create three synchronization channels. The signal that the car is travelling is sent by the automobile's automaton to the street's automaton, and it means that an automobile starts to move in the direction of the street. When sensor Si1 is activated, the automaton street sends and activates the channel *approach* to the automaton automobile, and it means that one automobile is at the street's start changing the automaton street to **OCCUPIED**. When the sensor So1 is activated, the automaton street sends and activates the channel *leave*, and it means that the automobile travelled the entire street and now is out of the street end. The street is again without any automobile and ready to receive another automobile.

Figure 3 shows the automaton street with the synchronization channels and respective places.

An automobile only has two possible states: *STOPPED* or *MOVING*. Sensors at the beginning and at the end of the street detect whether the automobile is in or out of the traffic lane. At the beginning of the simulation, the automobile is stopped, and the channel *travelling* will put the automobile moving in the direction of the start of the street, which has the sensor Si1. After being detected at the beginning of the street by the sensor Si1, the automobile will continue its trip until the end of the traffic lane. In order to create a permissive model, the automobile, after it finishes its trip in a street, can be forced to stop and go again in the direction of the street start or can continue moving in the direction of the street start.

Figure 4 represented the automaton automobile with the synchronization channels and the two places previously explained.

In this model, time features are not included and is only the first model created to represent the simplest interaction between an automobile and a street. Once, interactions between vehicles are not considered, the concept of neighbourhood is not applicable. The street is not discretized in cells, and this is the main reason for the absence of interactions between vehicles. If several automobiles were travelling in this model, its spatial position inside of the street would not be defined. The transition rule is that only one automobile can circulate in a street.

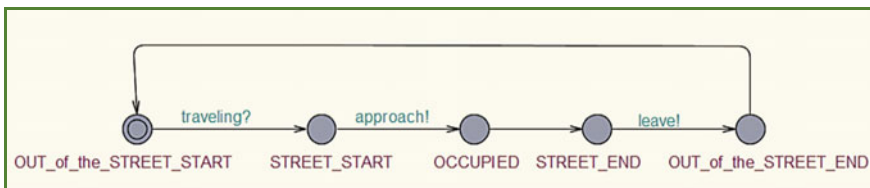
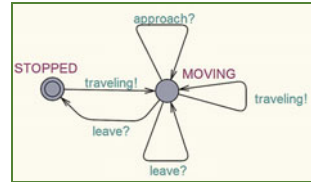


Fig. 3 Street automaton of the model with one automobile and one traffic lane

Fig. 4 Automaton *automobile* for the model with one automobile and one traffic lane



This model has the following limitations:

- It is a very simplified model, one only considers a type of vehicles (automobiles) and only one automobile can be inside of the street travelling.
- To implement in several streets with several automobiles, one would need to create each automaton automobile and street. Extending a scenario with a large number of automobiles and streets is complex.

Model with One Traffic Lane and Several Automobiles

In this new definition of the modelling problem, the complexity increases: one street with several automobiles moving inside it and interacting is simulated. To solve this problem two models were created. The first model presents the interaction between the vehicles moving cell by cell inside of the street if the cell in front of it is free using only variables to coordinate the movement. In the second model the interaction between the automobiles creates a queue at the end of the street if is not possible for them to circulate freely the entire length of the street using functions and variables to coordinate these movements.

Figure 5 schematically presents the physical environment of the model for one traffic lane and several automobiles. The automobiles are allocated inside of the street in the same order that initiated the movement.

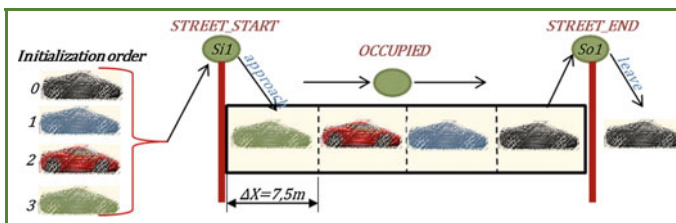


Fig. 5 Single traffic lane with several travelling automobiles

Model Using Only Variables to Coordinate the Interactions Between the Automobiles Travelling in a Traffic Lane

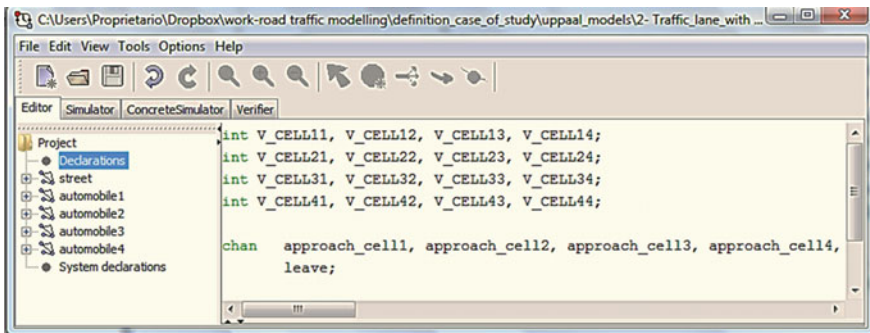
First, four control variables for each automobile were globally declared. The first digit of each variable refers to the number of the automobile, and the second digit corresponds to the respective street cell. The variable `V_CELL11` is responsible when automobile Nr. 1 is in the first cell of the street; the variable `V_CELL12` is the variable dedicated to the automobile Nr. 1 when it is in the second cell of the street. Similarly, for the others automobiles, control variables dedicated for each street cell were created.

In this example, the street has four cells and four automobiles will circulate. To implement this scenario four automata automobile and one automaton street were added.

Five channels were generated. The channel `approach_cell1` is activated when an automobile enters into the first cell of the street; the channel `approach_cell2` is activated when an automobile approaches the second street cell; the channel `approach_cell3` is activated when an automobile enters into the third cell of the street; the channel `approach_cell4` is activated when an automobile enters into the fourth street cell. The channel `leave` is activated when an automobile leaves the fourth and the last cell of the street. The channel `travelling` present in the previous model was erased, and it is assumed at the beginning of the simulation that all of the automobiles start moving inside the street with some velocity.

Figure 6 presents the global declaration for the model with one street with several automobiles travelling controlled with variables.

The automaton street is a loop of channels, i.e., it only contains the channels to coordinate the movement of the automobiles. The conditions for an automobile to move are in the structure of the automata automobiles through a combination of control variables. The sequence of the automobiles that will initialize its movement



```

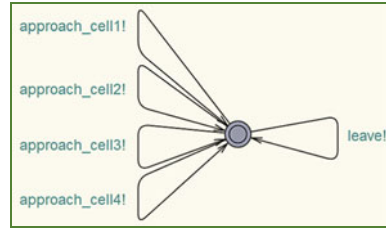
int V_CELL11, V_CELL12, V_CELL13, V_CELL14;
int V_CELL21, V_CELL22, V_CELL23, V_CELL24;
int V_CELL31, V_CELL32, V_CELL33, V_CELL34;
int V_CELL41, V_CELL42, V_CELL43, V_CELL44;

chan  approach_cell1, approach_cell2, approach_cell3, approach_cell4,
      leave;

```

Fig. 6 Global declaration for the model with one traffic lane with several automobiles, using only variables to control the movement

Fig. 7 Automaton *street* for the model with one traffic lane and several automobiles, using only variables to control the movement



can be implemented into the automaton street. The sequence of the automobiles was not considered.

Figure 7 represents the automaton street for the model with one street and several automobiles controlled only by variables.

The automaton automobile was reconfigured and has eight places:

- OUT_of_the_STREET_START_MOVING: When an automobile is moving in the direction of the beginning of the street
- CELL1: When an automobile has been detected by sensor S_{i1} , which means it is inside of the first street cell travelling.
- CELL2: When an automobile is travelling in the second street cell.
- CELL3: When an automobile is travelling in the third street cell.
- CELL4: When an automobile is travelling in the fourth street cell.
- WAITING_CELL1: When an automobile is waiting in the first street cell, until the second street cell becomes free again.
- WAITING_CELL2: When an automobile is waiting in the second street cell, until the third street cell becomes free again.
- WAITING_CELL3: When an automobile is waiting in the third street cell, until the fourth street cell becomes free again.

The bibliographic revision of Schreckenberg and Nagel in 1992 suggests a length of 7.5 m for each street cell. Therefore, the physical environment is a street with a total length of 30 m (four street cells).

An automobile travelling inside of the street will occupy one street cell when the model evolves. An automobile can only enter in the first street cell if all the others variables responsible for the first street cell from the other automobiles are equal to zero. This is the information declared in the guard of the first transition, meaning that the first street cell is free (Fig. 7). If this condition is verified, the automobile (in this case, automobile Nr. 4) can enter in the first street cell, which activates channel *approach_cell1*. At this time, the respective control variable for the first street cell of this automobile takes the value 1. With this variable updated, no other automobile can enter the first street cell, because the first street cell has an automobile. To continue to move, the automobile needs to verify if the second street cell is free. If this condition is true, it will move to the third street cell and update the respective control variable of the street cell Nr. 1 for the value zero and cell Nr. 2 for the value one. If the condition to move is not verified, the automobile evolves

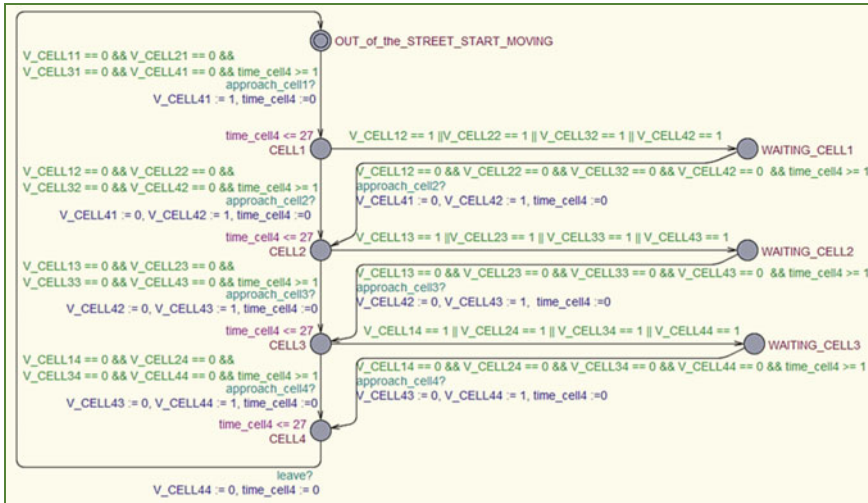


Fig. 8 Automaton automobile for automobile Nr. 4 is presented for the model with one traffic lane with several automobiles, using only variables to control the movement

to the place *waiting_cell1* and only when the cell in front is free can it move. The same procedure is executed for the followings cells.

When the automobile is in the last cell of the street, the channel *leave* is activated and the respective control variable of the last street cell is updated to the value zero, and the automobile leaves the street.

Figure 8 presents the automaton automobile for the model with one street with several automobiles only controlled with variables.

In this model, time conditions were already implemented. The time needed to travel one street cell meter with a range of velocities between 1 km/h and 50 km/h is 27 s and 0.5 s, respectively.

In this model, the physical environment was discretized in street cells with a length of 7.5 m, providing the groundwork for a neighbourhood. The neighbourhood of an automobile is the state of the street’s cell in front of it creating interactions between the automobiles. If the cell in front of it is occupied by another vehicle, it will wait until it is free again, and if the frontal street cell is free it can continue to move. The set of rules in this model is creating a queue of automobiles, from the beginning to the end of the street, by the same order that initialized its movements.

This model has the following limitations:

- It is a very simple model, because it considers only one type of vehicle (automobiles).
- To implement in several streets with several automobiles, it is necessary to create each automaton automobile and street and several control variables for each automobile responsible for each street cell; it is humanly impossible to

extend the model to a scenario with such a large number of automobiles and streets.

- If it is necessary to extend this model for intersections, the new upcoming street would be implemented in the automaton automobile, creating strict routes for each automobile, and the level of non-determinism would not be the desirable.
- For questions regarding organization, the features of the street (number of cells and conditions to move further) should be implemented in the automaton street and not in each automaton automobile.

Model Using Functions and Variables to Coordinate the Interactions Between the Automobiles Travelling in a Traffic Lane

In order to improve the limitations of previously presented models, a new model to solve the same modelling issues was created.

First, we declared in the global declaration the number of cells that the street contains. In this specific example, four street cells with a length of 7.5 m were declared, and this number of cells can be easily amended. Therefore, the physical environment created is a street 30 m long. The number of automobiles was declared using the function *typedef*. This function links and identifies each automobile with a specific integer. In this concrete model, four automobiles were declared. An automobile travelling inside the street occupies one street cell when the model evolves. For the interaction between the automobiles, five channels were created: *approach*, *leave*, *stop*, *stay* and *go*, which are functions of the number of automobiles present in the street. In this model also is assumed, at the beginning of the simulation, that all the automobiles start moving inside the street with some velocity.

Figure 9 presents the explained global declaration of the model one traffic lane with several automobiles controlled with functions and variables.

To simulate the desired behaviour and interactions between automobiles, it was necessary to create a new function and new auxiliary variables in the street declarations. Firstly, the automobiles are allocated for the same order that initialize their movement, in a list the same size as the number of the cells. This means that each element of the list is a street cell correlating to an element on the list with the *ID 0*, the last street's cell (end of the street) and the element with ID equal to the number of cells of the first street's cell (beginning of the street). A new variable *length* was declared, which is the total number of automobiles inside the street (length of the queue). For this reason, the *length* will have the maximum value equal to the number of street cells, because one automobile will occupy always one street cell.

The function *enqueue* was created to allocate the automobiles' IDs in the last cell of the street (list ID 0). This function will increase the value of the variable *length*

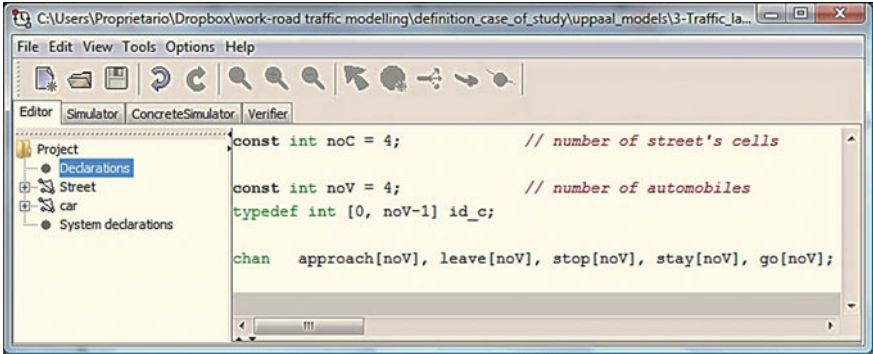


Fig. 9 Global declaration for the model with one traffic lane with several automobiles moving controlled by variables and functions

by one such that the synchronization channel *approach* is activated. If another automobile arrives while the previous is moving inside the street, the new automobile's ID are allocated in the second-to-last cell and the value of the variable *length* is equal to two until all the street's cells have an automobile inside. Consequently, the function *enqueue* is updated in the same arrow as the synchronization channel *approach*.

The function *dequeue* will release the last street's cell when the respective automobile inside it leaves the street and will decrease the value of the variable *length* by one. For this reason the function *dequeue* is associated and updated in the same arrow as the synchronization channel *leave*.

Another function created was the function 'front', which after an automobile leaves the street will move the vehicles to the next street's cell. This function is associated with the channel *go*.

The function *tail* was created to allocate an automobile that was forced to stop in the last position of the queue, and for this reason is associated with the synchronized channel *stop*.

Figure 10 presents the street declarations and the functions and new variables created.

The automaton street is more complex and has three places:

- **STREET_START**: When the sensor *Si1* has detected an automobile at the beginning of the street and the street is without an automobile.
- **OCCUPIED**: When one or more automobiles is inside of the street travelling.
- **OUT_of_the_STREET_END**: When an automobile was detected by the sensor *So1* at the end of the street and it leaves the street.

At the beginning of the simulation, this model assumes that the street is empty, without any automobiles moving inside. For this reason, the evolution from the first place *STREET_START* to the place *OCCUPIED* means that an automobile is travelling inside of the street. The function *enqueue* and the variable *length* are

```

id_c list[noC];
int[0,noC] length;

void enqueue(id_c element) // Put an automobile at the end of the queue
{
    list[length++] = element;
}

void dequeue() // Remove the front automobile of the queue
{
    int i = 0;
    length -= 1;
    while (i < length)
    {
        list[i] = list[i + 1];
        i++;
    }
    list[i] = 0;
}

id_c front() // Returns the front automobile of the queue
{
    return list[0];
}

id_c tail() // Returns the last automobile of the queue
{
    return list[length - 1];
}

```

Fig. 10 Street's declaration for the model with one traffic lane with several automobiles controlled by variables and functions

updated. In the location *OCCUPIED* the automobile can interact with others. For example, other automobiles can enter into the street and this automobile can be forced to stop and then start again its movement coordinated by the street's condition.

The automaton street only evolves to *STREET_END* when an automobile has been detected by the sensor *So1* and is leaving the street. If the street is occupied, it means that the length is greater than zero and the automaton returns to the place *OCCUPIED*. If the street is empty (length equal to zero) the automaton returns to *STREET_START*.

Figure 11 shows the configuration of the automaton street for the model with one traffic lane with several automobiles travelling and coordinated with variables and functions.

The automaton automobile was reconfigured and has five places:

- *OUT_of_the_STREET_START_MOVING*: When an automobile is moving towards the beginning of the street
- *MOVING_INSIDE_the_STREET*: When an automobile has been detected by sensor *Si1* and is inside of the street travelling.

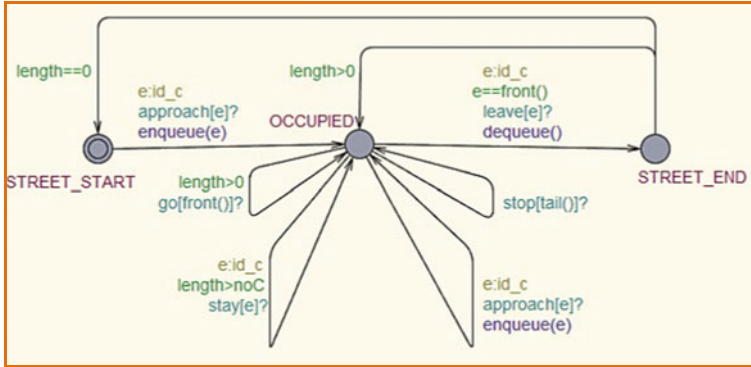


Fig. 11 Street automaton for the model with one traffic lane with several automobiles controlled by variables and functions

- STOPPED_INSIDE_of_the_STREET: When the automobile has stopped inside of the street.
- AGAIN_IN_MOVEMENT: When an automobile starts to move after a stop inside of the street.
- OUT_of_the_STREET_END_MOVING: When an automobile has been detected by sensor S_{01} exiting the street.

The automaton automobile only evolves from the initial location if it has been sensor S_{11} detects an automobile at the beginning of the street. From the moment an automobile is inside of the street, it has two possibilities of evolution: cross the entire street without a stop or is forced to stop. If the automobile reaches the end of the street without a stop, the time needed to travel four street cells with velocities between 1 km/h and 50 km/h is 108 s or 2 s, respectively. An automobile is forced to stop if there is another automobile in front of it travelling more slowly than it is or if the time allowed was more or less than the time needed to travel four street cells.

If an automobile is stopped on the street, it will only move if street conditions allow it to go in front. These evolutions in the automobile automaton are made by the synchronization channel *go*. If after that the street is free until the end, the automobile will reach the end of the street, but if it is forced to stop the channel *stay* is activated and the automobile stops again.

Figure 12 shows the configuration of the automaton automobile for the model with one street several automobiles.

In this model, similar to the previously presented model, the physical environment was discretized in street cells with a length of 7.5 m and the concept of neighbourhood is implemented. The neighbourhood of an automobile is the state of the street’s cell in front of it creating interactions between the automobiles.

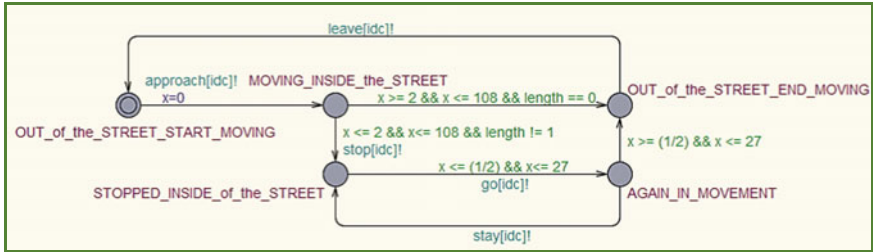


Fig. 12 Automaton automobile for a model with one street several automobiles controlled by variables and functions

The set of rules in this model creates a queue of automobiles at the end of the street if another automobile is travelling in the street or if the street is empty its entire length.

This model has the following limitations and advantages:

- It is a very simple model; it considers only one type of vehicles (automobiles).
- The number of street cells and automobiles can be easily changed; it is only necessary to define the global declaration of those variables.
- To implement it, several streets with several automobiles are created for each street automaton and several variable lengths and a list to allocate the automobiles in each street. It is humanly impossible to extend the model to a scenario with a large number of streets.
- The structure of the automaton street does not allow for the creation of streets with different lengths, because the time needed to travel all the streets is always the same.

The creation of these low complexity models was useful for understanding which modelling technique and structure can be used in order to reach the versatility required. At the end of this subchapter, we present an analysis of the two previous ways of modelling and the following conclusions can be verified:

- The number of vehicles and streets needs to be declared using the function *typedef*.
- One needs to rearrange the structure of the automaton automobile and street.
- To model the physical environment, neither changing the function list nor creating variables are good solutions for coordinating movements. In the case of the function list, the accuracy of the movement is not predictable, and this function it will create problems for calculating the crossing time of each road. It is not possible to extend the model for large scenarios by controlling by variables, because of the number of variables that are needed to implement into the automaton automobile.
- The structure to model the physical environment also needs to be easily changeable and accurate for the introduction of the traffic elements.

With these modelling requirements, we created a matrix of coordinates for modelling the physical environment. This matrix can control accurately the movement of the vehicles, precisely introduce traffic elements and is easily extendable by adding more rows. To model the physical environment of the streets matrices were used in all of the following models.

High Complexity Models

The basic models previously proposed, if applied to models with high levels of complexity or large traffic scenarios, cannot be easily be changed or reused, due the limitations previously explained.

In order to solve these limitations, models based upon the matrices were created to model the traffic environment. A matrix, which is a map of street cells, is a compact data structure that can easily be extended.

The matrices implemented to model the road traffic environment have the following features:

- Each line corresponds to an independent traffic lane of a street.
- The value present in column Nr. 1 corresponds to the number of cells that each street has.
- The following columns are filled with empty street cells or traffic elements.

Due to UPPAAL's limitations, one cannot create vectors with different sizes, and some of the values present in this matrix have no significance.

Traffic Lane with Multiple (1, 2, or 3) Possible Traffic Lanes Travelled by Automobiles in a Free Flow

As the behaviour of a street with several automobiles has already been modelled, the next level of complexity entails extending the map. Thus, at the end of the street upon which the automobiles started its movement, the automobiles will have to choose between one of three streets to continue their trips.

The rule to choose the next street should be based upon statistical data received by sensors implemented at the end of the streets. The data received give the traffic flow's percent for the upcoming streets. If the automobile at the end of the street has two possible next streets; the flow is distributed equally by the two streets. It means that 50% of the automobiles continue travelling on one street and 50% continue their movement on the other next street. When a street ends with three possible next streets, the traffic flow will also be distributed equally. Consequently, each next street will have 33% of the traffic flow circulating on the previous street.

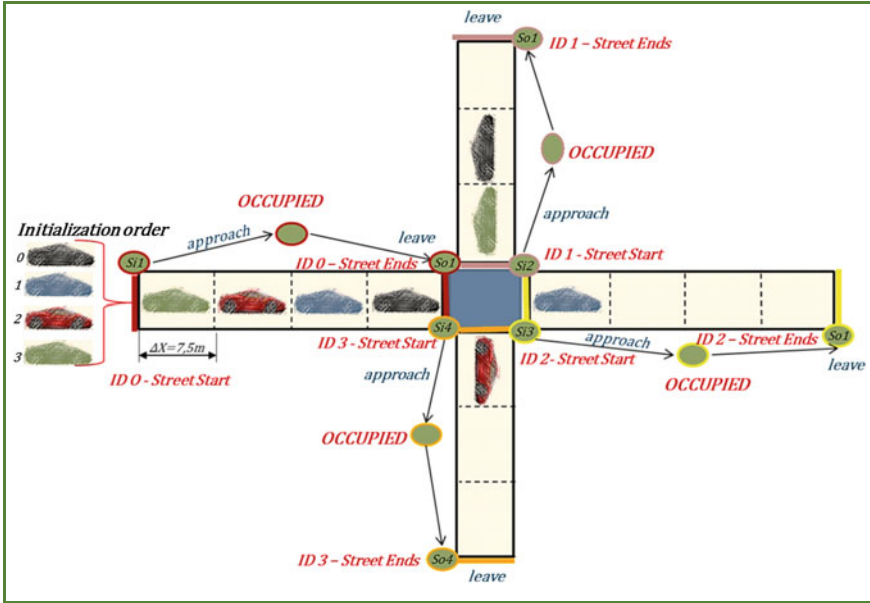


Fig. 13 Schematic diagram of the model with a traffic lane with several automobiles travelling with three possible next streets

Figure 13 presents the model with several automobiles moving in a street, and at the end of the street, another street is chosen. The previous paragraph explained the distribution of the traffic flow with the rules admitted represented in this scheme. The first automobile (black), which arrives in the intersection (blue rectangle), travels on the street ID 1, the second one to reach the intersection (blue) will choose the street ID 2, the third one (red) will choose the street ID 3, and this cycle of choice will continue. This distribution of the traffic flow seems to be reasonable and to create a model more accurate than taking information from each traffic lane that is studied.

The authors have long been concerned with the creation of general models that can be easily extended to or implemented in new situations and features. For this reason, the previous definition of the street cell using the function list was erased and in this new model, similar to the automobiles' definition, we used the function *typedef* to create the streets in the global declarations.

To define the physical environment, we created another variable, *maxnoCells*. This variable defines the maximum length of the street considered (maximum number of cells). With this new variable and with the variable number of streets (noS), we created a matrix *idexSC*. This matrix is a map of street cell coordinates. Each line corresponds to a street ID with the first line ID0 and the last line the last street's ID. The column Nr. 1 corresponds to the number of cells that each street has and the following columns are filled with -1, which means a street's cell is empty.

Due to UPPAAL's limitations, it is not possible to create vectors with different sizes, thus many of the -1 s present in the matrix have no significance. The first column is important for limiting the street size. Therefore, it is easy to extend the map and to implement other features in the posterior models.

The choice of the next street in an intersection was implemented with a similar application. First, a new variable, *maxNextStreets*, (maximum next streets) was declared. This variable is equal to three, because at a maximum, an automobile has three possible next streets in this intersection. With this variable and with the variable number of the streets (noS), we created another matrix, *indexMAP*. Each line corresponds to a street ID with the first line the street's ID 0 and the last line the last street's ID. The first column contains the number of possible next streets at the end of the street, and the other columns have the street IDs of the next streets. As presented in the scheme of Fig. 12, the street ID 0 at its end has three possible next streets (street ID 1, 2 or 3), and all the automobiles that are in the last cell of the street IDs 1, 2, or 3 have zero possible next streets. Thus when an automobile leaves the street IDs 1, 2 or 3 are considered to have left the map. The value -1 means that there is no upcoming street, and the automobile will exit the map. The variable b is a control variable, a function of the number of the street. For each value of b an automobile present at the end of the street will have another destination.

The current street for each automobile has been defined, and the value defined in the vector *currentStreet* represents the initial street's ID that an automobile will start the simulation. This vector contains the inputs of the traffic flow in the simulation. The value of *currentStreet* will change every time that an automobile leaves a street and continues to move in another street of the map.

All the channels present are a function of the number of automobiles. The channel approach is activated when a vehicle is detected at the beginning of the street (first street's cell), the channel *travelling* is activated when an automobile has a free cell in front of it of it, and the channel *leave* is activated when an automobile is in the last street's cell.

In Fig. 14 the global variables are presented in the model with one traffic lane and multiple possible choices as upcoming next streets with automobiles moving in a free flow.

The automaton street was simplified, because there are only three channels but continues with three places:

- **STREET_STARTS**: When a sensor S_{i1} has detected an automobile at the beginning of a street and there are no automobiles on the street.
- **UPDATING_CELLS**: When one or more automobiles are travelling on a street.
- **STREET_ENDS**: When the sensor S_{o1} has detected an automobile at the end of the street and the automobile exits the street.
- This model assumes at the beginning of the simulation, that the entire map of the street is empty, without any automobiles moving inside it. For this reason, the evolution from the first place *STREET_START* to the place *UPDATING_CELLS* means that an automobile is travelling on a street and updates the function *enqueue*. To be safe, verify three rules: First, read the value inside of the vector

```

const int noA = 4;           // Number of AUTOMOBILES
typedef int [0, noA-1] idA; // Automobiles' ID
//-----
const int noS = 4;           // Number of STREETS
typedef int [0,noS-1] idS;  // Streets' ID
//-----
const int maxnoCells = 6; // Maximum number of cells (maximum street lenght)

// Description of the matrix number of cells per different street
int indexSC[noS][maxnoCells+1] = {{6,-1,-1,-1,-1,-1,-1},
                                   {5,-1,-1,-1,-1,-1,-1},
                                   {6,-1,-1,-1,-1,-1,-1},
                                   {4,-1,-1,-1,-1,-1,-1}};
//-----
// Maximum number of possible streets for a car in the last cell of a street
const int maxNextStreets = 3;

//Description of the matrix street possibilities
int indexMAP[noS][maxNextStreets+1] = {{3, 1, 2, 3},
                                         {0,-1,-1,-1},
                                         {0,-1,-1,-1},
                                         {0,-1,-1,-1}};

// auxiliar variable to control the choice at the street ends
int b[noS] = {0,0,0,0};
//-----
// variable which contains the street's ID for each car at the beginning of the simulation
int currentStreetA[noA] = {0, 0, 0, 0};
//-----
chan  approach[noA], leave[noA], traveling[noA];
    
```

Fig. 14 Global declarations of the model traffic lane with several automobiles travelling with three possible next streets

currentStreet, which will define the street where an automobile is arriving. Second the matrix *indexSD* needs to have in its second column the value -1 , which means that the first street's cell is empty without any vehicle. Third, the variable *novis* (number the vehicles inside of the street) needs to be smaller or equal to the number of cells, which means at maximum each street can only contain an number of automobiles equal to the number of street cells, because each automobile will occupy one cell.

In the location *UPDATING_CELLS*, the automobiles can interact with each other. For example, another automobile can enter the street with the same rules described previously and another automobile can continue to travelling, if the number of the cells travelled (*nocTA*) is smaller than the size of the street (smaller than the first column of the matrix *indexSC*) and if the cell in front of it is free (the cell needs to have the value -1). It is necessary to read the value present in the *currentStreet* vector to know the street on which an automobile is moving. In this second case, the channel *travelling* and function *moving* are updated.

The automaton street only evolves from the place *UPDATING_CELLS* when the sensor *S01* detects an automobile leaving the street. In this case, the variable *nocTA*

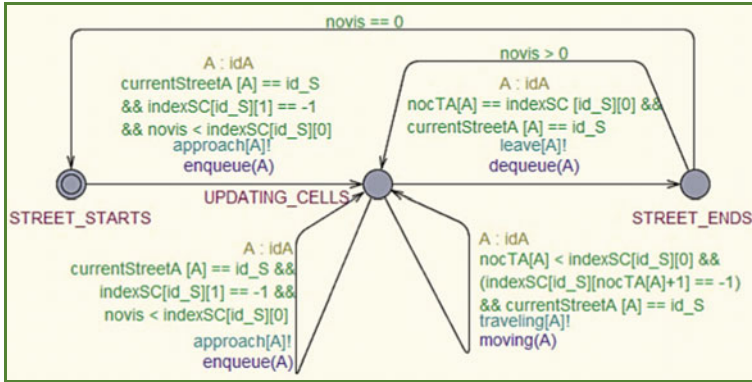


Fig. 15 Street automaton of the model traffic lane with several automobiles travelling with three possible next streets

is equal to the first column of the matrix *indexSC*. To know the street that an automobile is leaving, it is also necessary to read the value present in the vector *currentStreet*. If the variable *novis* is greater than zero, the street automaton returns to the place *UPDATING_CELLS*, because the street is not empty and has automobiles travelling. Another scenario is if the number of the automobiles on the street is equal to zero, the automaton returns to the place *STREET_START* with the street empty.

The automaton street is the “brain” of the simulation having all the information regarding traffic conditions, a set of rules and consequent transition rules.

Figure 15 shows the configuration of the automaton street for the model with one traffic lane and several automobiles travelling with three possible next streets.

In street declarations, three functions were implemented (*enqueue*, *dequeue* and *moving*), which will contain a set of rules for the correct traffic circulation.

The *enqueue* function defines rules for an automobile that enters a street. In a real scenario an automobile can only enter a street if the street is not full of vehicles and starts its trip from the beginning of the street until the end.

The function *moving* contains the rules for an automobile continues to move. In the reality an automobile can only move further if there is free space in front of it.

The function *dequeue* comprehends the rules for an automobile leaving a street. The function *dequeue* will verify if the automobile is in the last street’s cell, and if this condition is true, the automobile exits the street.

Figure 16 presents the explained street declarations and the created functions and variables. The street declarations are the places where all the transition rules for a street are implemented.

The automaton automobile was reconfigured and has three places:

- **OUT_of_the_STREET_START_MOVING**: When an automobile is moving towards the beginning of the street.

```

int nocTA[noA] = {1,1,1,1}; // number of cells travelled by an automobile
int novis ; // number of vehicles inside of the street

void enqueue(idA automobile) // Put an element (AUTOMOBILE) in the first cell of the street
{
  // if the first cell of the street is free and novis is smaller than first column of indexSC
  if ((indexSC[id_S][nocTA[automobile]] == -1) && (novis < indexSC[id_S][0]))
  {
    indexSC[id_S][nocTA[automobile]] := automobile; // add an element in the first cell
    novis++;
  }
}

void moving (idA automobile) // Moving an element (AUTOMOBILE) inside of the street
{
  // if the cell in front is free and nocTA is smaller than first column of indexSC
  if ((indexSC[id_S][nocTA[automobile]+1] == -1) && (nocTA[automobile] < indexSC[id_S][0]))
  {
    indexSC[id_S][nocTA[automobile]+1] := automobile; // add an element in the cell in front
    indexSC[id_S][nocTA[automobile]] := -1; //release the previous cell
    nocTA[automobile]++;
  }
}

void dequeue(idA automobile) // Remove the front element (AUTOMOBILE) of the street
{
  // if nocTA is equal to the first column of indexSC
  if (nocTA[automobile] == indexSC[id_S][0])
  {
    indexSC[id_S][nocTA[automobile]] := -1; //release the last cell of the street
    nocTA[automobile] := 1; // reset the number of the cells for the initial value
    novis--;
  }
}

```

Fig. 16 Street declarations for the model with a traffic lane with several automobiles travelling with three possible next streets

- **MOVING_INSIDE_the_STREET**: When the sensor Si_1 has detected an automobile, and the automobile is travelling on the street.
- **OUT_of_the_MAP**: When the sensor So_1 detects the automobile, and there are no possible next streets, they are considered out of the map.

Figure 17 shows the configuration of the automaton automobile for the model with one traffic lane and several automobiles travelling with three possible next streets.

This model is the first model with relevant traffic road rules included and a significant level of detail. At the end of this subchapter, we briefly present some features of this model:

- The physical environment is a one-dimensional grid of rectangular cells, all equal in size (7.5 m of length).
- It is a single cell model, because each automobile occupies only one street cell in each time iteration, and the cells can only have two possible states: occupied by an automobile or empty.
- The size of the neighbourhood is the same for each cell. The model is anisotropic, because the automobiles only respond to stimulus in front of it.

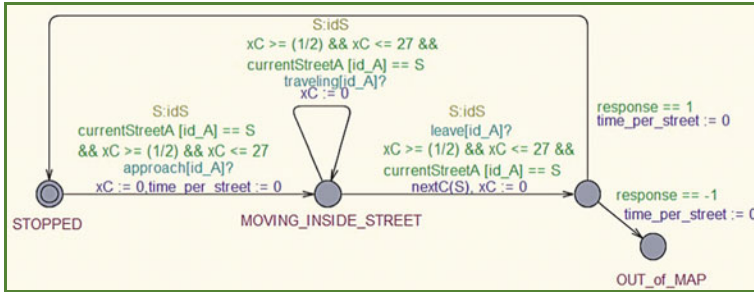


Fig. 17 *Automobile* automaton for the model traffic lane with several automobiles travelling with three possible next streets

- This model is a dynamic system with a closed number of automobiles and streets. The evolution in space and time depending on the same rules. Only if the cell in front is free, can the automobile proceed on the street.
- The time is a stochastic feature and its choice is completely non deterministic. In an instant t , the automobile can circulate in a cell at velocity of 50 km/h and in the instant $t + 1$, it can move at 2 km/h. This model presents heavy breaking and quick accelerations, even if the street is completely free.
- It is a very simplified model; one only considers a type of vehicles (automobiles).
- This model can easily be extended.

In fact, this model has the potential to be extended to new traffic scenarios and be implemented with other types of vehicles and traffic elements. Even in those scenarios, this model provides a solid basis.

In addition, one can consider other approaches and behavioural properties [31, 32] for the whole system.

Conclusions

The cellular automata allow for the observation of different phenomena, dissecting components into individual variables, allowing one to understand how local changes affect the whole grid of cells.

The formalism-timed automata are appropriate for a modular approach due to its elementary structure. The resolution (level of detail) and system size (the network size to be covered) obtained are appropriate for the proposed model.

In the context of urban traffic theory, the cellular automata in microscopic models have the capacity to simulate in detail all the elements presented in this traffic environment. The quantities of traffic elements implemented generate a model containing a large number of evolutionary rules and interactions.

The main goal of this work was the creation of a systematic approach for a complex urban traffic scenario. The structure implemented has the potential to be expanded. This systematic approach can be easily and limitlessly extended, the only limitation being the computational power available.

This model has flexibility in accordance with the environment that it is applied, because it can implement a large group of traffic elements and possible interactions. The stimuli created in each vehicle depend on the traffic elements contained in the street and the traffic conditions, because of the modular approach.

As future work, the authors will implement the distributed control of an illustrated case study considering some issues and solutions used on industrial networks [33–38] and adapted to this field of application. An analysis of performance of the developed distributed controller will also be considered [39, 40].

References

1. Clark J, Daigle G (1997) The importance of simulation techniques in its research and analysis. In: Andradóttir S, Healy KJ, Withers DH, Nelson BL (eds) Proceedings of the 1997 winter simulation conference, USA
2. Lima EB (2007) Modelos microscópicos para simulação do tráfego baseados em autômatos celulares. Universidade Federal Fluminense, Niterói, Brazil, Dissertação de Mestrado em Computação
3. May AD (1997) Traffic flow fundamentals. Prentice Hall
4. Vico FJ, Basagoiti FJ, Platas RG, Lobo D (2005) Modelado y simulación del tráfico en vías urbanas y periurbanas en base a la estimación de tiempos de recorrido. Universidad de Málaga and Tecnologías Viales Aplicadas, TEVA, SL, Malaga, ETSI Informática
5. Nagel K, Schreckenberg M (1992) A cellular automaton model for freeway traffic. *J de Phys I, France*, 2221–2229
6. Nakanishi K, Itoh K, Igarashi Y, Bando M (1996) Solvable optimal velocity models and asymptotic trajectory. Department of Physics, Kyoto University, Kyoto 606-01, Physics Division, Dept. of Education, Niigata University, Niigata 950-21, and Physics Division, Aichi University, Miyoshi, Aichi 470-02, 1996, Japan
7. Chowdhury D, Santen L, Schadschneider A (2000) Statistical physics of vehicular traffic and some related systems. *Phys Rep* 329:199–329
8. Chua L (2005) A nonlinear dynamics perspective of wolfram’s new kind of science. In: Bernoulli shift to universal computation, inaugural lecture of the international Francqui chair, Katholieke Universiteit Leuven, June, 2005
9. Murata T (1989) Petri nets: properties, analysis and applications. In: Proceedings of the IEE, vol 77, no 4. Department of Electrical Engineering and Computer Science, University of Illinois, Chicago, USA Apr 1989
10. Maerivoet S, Moor BD (2005) Transportation planning and traffic flow models. 05–155, Katholieke Universiteit Leuven, Department of Electrical Engineering ESAT-SCD (SISTA), July 2005
11. Maerivoet S, Moor BD (2005) Cellular automata models of road traffic. In: Physics reports (ed) Department of Electrical Engineering ESAT-SCD (SISTA), Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium, 12 Sept 2005
12. Wolfram S (1983) Statistical mechanics of cellular automata. *Rev Mod Phys* 55:601–644
13. Wolfram S (2002) A new kind of science. Wolfram Media, Inc., ISBN 1-579-955008-8

14. Wang J Petri nets for dynamic event-driven system modeling. Department of Software Engineering, Monmouth University, West Long Branch, USA
15. Nagel K (1996) Particle hopping models and traffic flow theory. *Phys Rev E* 53(5):4655–4672
16. Rothery R (1998) Traffic flow theory. Transportation Research Board (TRB). Special Report, p 165
17. Gardner M (1970) Mathematical games—the fantastic combinations of John Conway’s new solitaire game “life”. *Sci Am*, 120–123
18. Milner R (1989) *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs
19. Alur R, Dill DL (1994) A theory of timed automata. *Theor Comput Sci Elsevier* 126:183–235
20. Angulo FRF (2014) *Autómata celular*
21. Eppstein D (2014) *Cellular automaton*
22. Vaandrager F A first introduction to Uppaal—a job shop example. Institute for Computing and Information Sciences, Radboud University Nijmegen, Heijendaalseweg 135, 6525 AJ Nijmegen, Netherlands
23. Machado J, Seabra E, Campos JC, Soares F, Leão CP (2011) Safe controllers design for industrial automation systems. *Comput Ind Eng* 60(4):635–653
24. Moreira A (2003) Universality and decidability of number-conserving cellular automata. *Theor Comput Sci* 292:711–721
25. Crutchfield JP, Kaneko K (1987) Phenomenology of spatiotemporal chaos. In: *Directions in chaos*. World Scientific, pp 272–353
26. Behrmann G, David A, Larsen KG (2004) A tutorial on UPPAAL. In: *Proceedings of the 4th international school on formal methods for the design of computer, communication, and software systems (SFM-RT’04)*. LNCS 3185
27. Kaneko K (1990) Simulating physics with coupled map lattices. In: Kawasaki K, Onuki A, Suzuki M (eds) *Formation, dynamics, and statistics of patterns*. World Scientific, pp 1–52
28. Kier LB, Seybold PG, Cheng C-K *Cellular automata modeling of chemical systems*. Published in Springer (ed), Center for the study of biological complexity. Virginia Commonwealth University, Richmond Virginia, USA
29. Kunz G, Machado J, Perondi E (2015) Using timed automata for modeling, simulating and verifying networked systems controller’s specifications. *Neural Comput. Appl.*, 1–11
30. Machado J, Denis B, Lesage J-J (2006) A generic approach to build plant models for des verification purposes. In: *Proceedings—eighth international workshop on discrete event systems, WODES 2006*, pp 407–412
31. Campos AM (2014) *Autómata Celular*
32. Campos JC, Machado J, Seabra E (2008) Property patterns for the formal verification of automated production systems. In: *IFAC proceedings volumes (IFAC-PapersOnline)*, 17 (1 Part 1)
33. Barros C, Leao CP, Soares F, Minas G, Machado J (2013) Issues in remote laboratory developments for biomedical engineering education. In: *International conference on interactive collaborative learning, ICL 2013*, art. no 6644585, pp 290–295
34. Costa J, Carvalho N, Soares F, Machado J (2009) The fins protocol for complex industrial applications: a case study. In: *ICINCO 2009—6th international conference on informatics in control, automation and robotics*. Proceedings, 2 RA, pp 348–354
35. Leão CP, Soares FO, Machado JM, Seabra E, Rodrigues H (2011) Design and development of an industrial network laboratory. In: *Int J Emerg Technol Learn* 6(Special Issue 2):21–26
36. Leão CP, Soares F, Rodrigues H, Seabra E, Machado J, Farinha P, Costa S (2012) Web-assisted laboratory for control education: remote and virtual environments. *Communications in Computer and Information Science*, 282 CCIS, pp 62–72
37. Silva M, Pereira F, Soares F, Leão CP, Machado J, Carvalho V (2015) An overview of industrial communication networks. *Mech Mach Sci* 24:933–940
38. Silva PCM (2001) *Teoria do fluxo de tráfego*. Universidade de Brasília, Brasil, Material didático do curso de engenharia de tráfego

39. Ceccarelli M, Carbone G, Ottaviano E (2005) Multi criteria optimum design of manipulators. *Bull Pol Acad Sci Tech Sci* 53(1):9–18
40. Thomas F, Ottaviano E, Ros L, Ceccarelli M (2005) Performance analysis of a 3-2-1 pose estimation device. *IEEE Trans Rob* 21(3):288–297