ELSEVIER
SCIENCE

A Modeling Framework for Urban Traffic Systems Microscopic Simulation

Emmanuel López-Neri, Antonio Ramírez-Treviño, Ernesto López-Mellado*

CINVESTAV Unidad Guadalajara, Av. Científica 1145 Col El Bajío, 45015 Zapopan Jal. Mexico

Elsevier use only: Received date here; revised date here; accepted date here

Abstract

In this paper a modeling framework for urban traffic systems (UTS) is presented. The model, used for agent based micro-simulation, describes both the traffic network and dynamic entities, namely vehicles, traffic lights, and pedestrians. The framework allows defining systematically the necessary components and their behavior of a model oriented to event driven simulation, which can be executed in a distributed way. In the model, the vehicles are conceived as mobile agents with decision making capabilities that interact with the environment and other entities within the traffic network, performing diverse activities according to numerous situations arisen during the simulation. A multi-level Petri net based formalism, named n-LNS is used for describing the structure of the UTS and the components behavior. The first level describes the traffic network; the second level models the behavior of diverse road network users considered as agents, and the third level specifies detailed procedures performed by the agents, namely travel plans, tasks, etc.

Keywords: Urban traffic microsimulation; Modeling framework; Mobile agent systems

1. Introduction

The urban traffic systems (UTS) are complex systems composed of vehicles, pedestrians, traffic lights, a traffic network structure and some sub-systems (Urban Traffic Management Control or Population Generation). The large number of the vehicles provokes well known problems such as traffic jams, air and noise pollution, fuel consumption, stress to drivers, etc. These problems may be reduced by the efficient use of current urban resources through the performance analysis under different traffic light control policies along the day. Model-based simulation is often used for evaluating UTS yielding statistics about travel times, fuel consumption, and road density; such information is useful to analyze traffic control strategies, urban transport routes, etc.

In the literature there exist different approaches to model UTS. In [1] UML is used to describe in a hierarchical way the microscopic and macroscopic information, traffic control policies, and network structure relations. In [2] the pattern designs from the object oriented programming paradigm is used to capture the UTS structure and entities relationships. Although these models reduce the complexity design, they do not allow evaluating the correctness and completeness of the obtained models. In [3] the hybrid Petri net formalism is used to represent in a modular way the UTS microscopic and macroscopic information; however the decision making level cannot be directly represented. Cell-DEVS is another formal tool described in [12], which allows describing formally a system using event-oriented or time-oriented approach. In [5] Cell-DEVS is used for evaluating intelligent

* Corresponding author. Tel.: +52-33-37773600; fax: +0-000-000-0000 ; e-mail: author@institute.xxx .

transportation systems (ITS). Cell-DEVS is based on cellular automata to represent the movement of the objects; however it has been shown in [10] that cellular automata are not suitable to represent moving objects yielding modeling errors introduced by the users when synchronous updating is required.

The micro simulations are more realistic when the road and user behavior are described based on the multi-agent paradigm. In [14] a reactive, autonomous and social multi-agent model to describe the vehicle behavior is presented. In [15] an agent model for pedestrians' behavior using automata cellular is presented. However these models use a time-slicing approach for updating the agents, which highly consume computational resources and decrease the number of agents interacting in the environment. In fact, the multi-agent paradigm is used to model management strategies rather than to describe vehicle driver behaviors [16].

In [17] a meta-model framework is presented in which the streets are described as agents and the cars are messages sent and consumed by streets agents. Since that approach does not consider the cars as agents, then the car behavior and other complex behaviors cannot be captured. There exist other approaches that do not represent the environment as a first class model component [18].

Our approach for microscopic simulation is agent based; dynamic entities (vehicles) evolving within the traffic network are implemented as mobile agents allowing representing drivers behavior and interactions with the environment. This paper focuses on the formal specification of knowledge and data required for developing a simulation engine and performing simulations. For this purpose a modeling framework allowing describing both the UTS components and their behavior is proposed. An UTS model is expressed through several data structures and a three level net system allowing validating the correctness and completeness of the UTS model.

The paper is organized as follows. Section 2 presents the n-LNS formalism. Section 3 presents UTS taxonomy. Section 4 describes the modeling framework. Section 5 presents the mechanisms for model execution. Section 6 presents a simulation case study.

2. The n-LNS Formalism

The formalism follows the approach of nets within nets introduced by R. Valk [24], in which a two level nested net scheme called EOS (Elementary Object System) is proposed. An extension to the Valk's technique, called n-LNS, has been proposed [23]; in this section we present an overview of n-LNS. A more accurate definition of the formalism is detailed in [23].

Definition

An n-LNS model consists mainly of an arbitrary number of nets organized in n levels according to a hierarchy; n depends on the degree of abstraction that is desired in the model. A net may handle as tokens, nets of deeper levels and symbols; the nets of level n permits only symbols as tokens, similarly to CPN (Colored PN). Interactions among nets are declared through symbolic labeling of transitions.

Fig. 1 sketches pieces of the components of a 4-LNS model. The level 1 is represented by the net NET_1, the level 2 by the nets $NET_{2,1}$ and $NET_{2,2}$, the nets $NET_{3,1}$, $NET_{3,2}$, NET3,3, and $NET_{3,4}$ compose the level 3, and the nets $NET_{4,1}$, $NET_{4,2}$, $NET_{4,3}$ form the level 4.

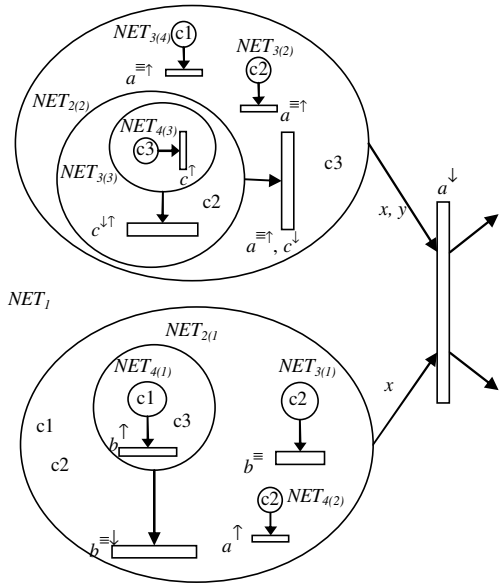


Fig. 1: Piece of a 4-LNS

A net of level *i* is a tuple $NET_i = (typenet_i, \mu_i)$, where is composed by a PN structure, the arcs weight ($\pi((p, t)$, *lab*) or $\pi((t, p)$, *lab*)) expressed as multi sets of variables and symbols, and a transition labeling function declaring the net interaction. μ_i is the marking function (see Figure 2).

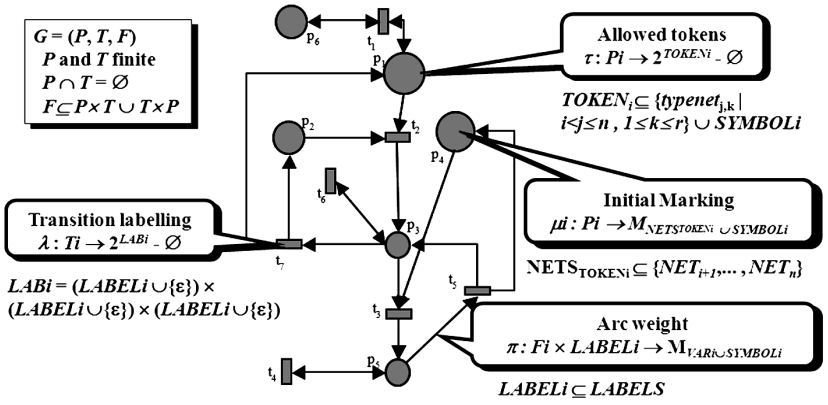


Fig. 2: Example of a Net of Level *i*, in the n-LNS Formalism, defined as $Net_i = ((typenet_i , \mu_i)$, where $typenet_i = (G, TOKEN_i, LABEL_i, VAR_i, \tau, \lambda, \pi))$.

A n-LNS model, called net system, is a n-tuple NS= (NET_1, NET_2, … NET_n) where NET_1 is the highest level net, and $NET_i = \{NET_{i,1} , NET_{i,2}, ... , NET_{i,r}\}$ is a set of r nets of level *i*.

The components of a model may interact among them through synchronization of transitions. The synchronization mechanism is included in the enabling and firing rules of the transitions; it establishes that two or more transitions labeled with the same symbol must be synchronized. A label may have the attributes ≡, ↓, ↑, which express local, inner, and external synchronization respectively.

Transition Enabling and Firing

A transition t of a net of level i NET_i is enabled with respect to a label lab if:

- There exists a binding b_t that associates the set of variables appearing in all $\pi((p,t),lab)$.
- It must fulfill that $\forall p \in {}^\bullet t$, $\pi((p, t), lab)_{<bt>} \subseteq \mu_i(p)$. ($<b_t>$ is not necessary when the level net is n).
- The conditions of one of the following cases are fulfilled:
  - *Case 1*. If there is not attributes then the firing of t is autonomously performed.
  - *Case 2*. If lab has attributes one must consider the combination of the following situations:
    - $\{\equiv\}$ It is required the simultaneous enabling of the transitions labeled with $lab^{\equiv}$ belonging to other nets into the same place p' of the next upper level net. The firing of these transitions is simultaneous and all the (locally) synchronized nets remain into p'.
    - $\{\downarrow\}$ It is required the enabling of the transitions labeled with $lab^{\uparrow}$ belonging to other lower level nets into $\bullet t$. These transitions fire simultaneously and the lower level nets and symbols declared by $\pi((p, t), lab)_{<bt>}$ are removed.
    - $\{\uparrow\}$ It is required the enabling of at least one of the $t' \in p'^{\bullet}$, labeled with $lab^{\downarrow}$, of the upper level net where the NET_i is contained. The firing of t provokes the transfer of NET_i and symbols declared into $\pi((p', t'), lab)_{<bt>}$.

The firing of transitions in all level nets modifies the marking by removing $\pi((p, t), lab)_{<bt>}$ in all the input places and adding $\pi((t, p), lab)_{<bt>}$ to the output places.

In Figure 1, NET_1 is synchronized through the transition labeled using $a^{\downarrow}$ with $NET_{2,2}$, $NET_{3,2}$, $NET_{3,4}$ and $NET_{4,2}$ by mean the transitions (locally synchronized) labeled with $a^{\uparrow}$; all these transitions must be enabled to fire. The simultaneous firing of the transitions removes these nets from the input places.

$NET_{2,1}$, $NET_{3,1}$ and $NET_{4,1}$ are synchronized through the transitions labeled with $b^{\downarrow}$, $b^{\equiv}$, $b^{\uparrow}$ respectively; the firing of the transitions changes the marking of $NET_{2,1}$ and $NET_{3,1}$; $NET_{4,1}$ is removed from the place of $NET_{2,1}$. $NET_{3,3}$ is removed from the input place of $NET_{2,2}$ and $NET_{4,3}$ is removed from $NET_{3,3}$; this interaction is established by $c^{\downarrow}$, $c^{\downarrow\uparrow}$, $c^{\uparrow}$, respectively.

3. Urban Traffic System Components

This section presents UTS taxonomy. It describes intuitively the relevant components of the physical system that should be captured in the formal model. To describe this taxonomy a UML based diagram is used and is depicted in Fig. 3. The first component we take into account is the road traffic network, which is composed of all the kinds of roads in an urban traffic system: streets, crosswalks, intersections, on/off ramps, etc., and of traffic signals that are used to impart traffic information. This last one could be classified as static and dynamic traffic signals. The static components cannot change their state, for instance, speed limit sign, priority traffic sign, etc., which are used to impart information. The dynamical components can change their state along time, for instance, traffic lights, variable messages signs, etc.

The second component is a set of mobile entities called road users, namely vehicles and pedestrians. The users behave according to evolving rules that govern their individual and joint behavior and, usually, the urban traffic policies conceived for improving the traffic safety and efficiency.

The interaction of one road user with other users, static components, and traffic information signals leads to more complex behaviors known as emergent behavior. For instance queues, traffic jams, gridlock, green wave, etc. are considered as emergent behavior.

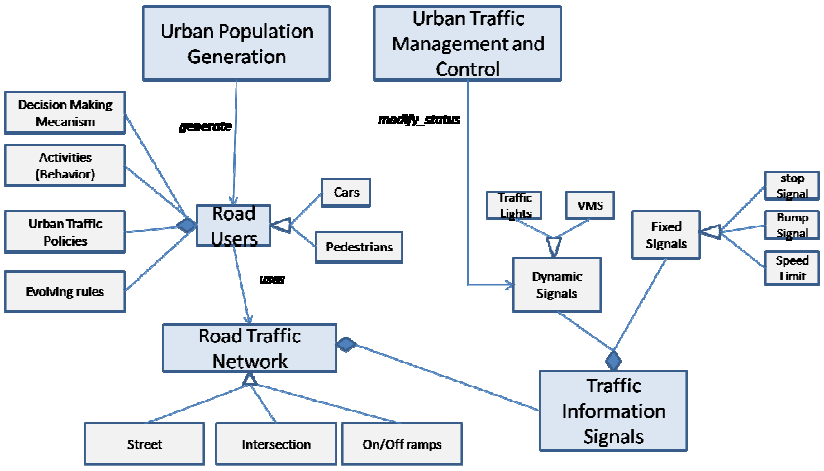


Fig. 3. Urban Traffic System Taxonomy

There exist some sub-systems of the UTS that can modify the state of some system entities or generate new entities into the system. The traffic lights are one of those entities that could receive messages from the traffic control system (TCS) in order to modify their state (for instance, the green light to red light, etc.). The urban population generation system, for instance, generates the road users based on some demographic information which is based in census and typical data such as gender and age, this sub-system also generates a list of activities (home, going shopping, going to work, etc.) and activity locations for each road user [20,21,22].

Thus the components of an urban traffic system are: road traffic network, road users (vehicle, pedestrians, cyclists, etc.), traffic signs (dynamic: traffic light, and static: speed limit sign), road traffic network, road traffic policies, evolving rules, environmental rules and some subsystems, for instance: demographic population generation, weather system, urban traffic control, etc.

4. Modeling framework

The proposed modeling framework is based on classifying the UTS entities into three categories: environment, actor and stimulus entities (see Fig. 4). Also a stimulus subsystems category is introduced for instance: urban traffic control, demographic info, etc. First, two main questions must be answered: What is the goal (objectives) of the simulation? and which are the problems desired to analyze?. The simulation goal is used to select the functional primitives (segments, objects, intersections and event trajectories) used to construct the environment and actor models, it means that it will allow to define the granularity level of the desired variables to be observed during the simulation. By using the n-LNS formalism, the environment, actor and stimulus models are constructed. Once all models (environment and actor) are obtained then the relationships among them are defined. Thus, the executable model is constructed and compiled. Next, a simulator kernel is used to implement the obtained models and get the resulting UTS simulator. Finally, are defined the input info for the stimulus subsystems (for instance, the control policies for the traffic lights, or the probability distribution for arrivals, etc.), the model is executed and a performance analysis of system is done.

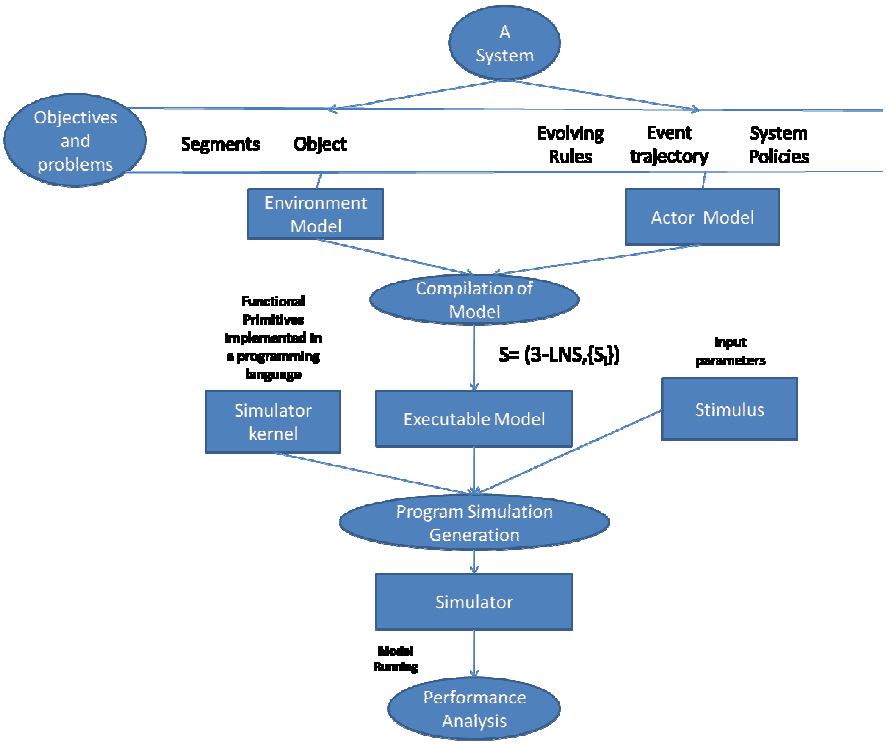


Fig. 4: Proposed Urban Systems Modeling Methodology

4.1. Mobile Actor Model

The road users are defined as mobile actors in the model. Mobile actors have individual goals and particular behaviors that need the system environment to achieve their goals, for instance, cars, buses, pedestrians, etc.

The actor behavior is constrained by evolving rules, urban traffic policies, infrastructure resources availability and the behavior of other entities and is described as the trajectory of events and their correspondent states along the time, a.k.a behavioral data [19]. Each time an actor event is fired, some of the actor attributes are modified, for instance: velocity, vehicle occupants, quality of service, position, etc. The relevant events for the "car" entity are: starting to accelerate, init to stop, starting to break, etc., all of this at its own preferred velocity. These events describe the start or end of some entity states, for instance: stopping, accelerating, breaking, advancing.

Since in a real UTS the car drivers see other car behavior in a limited neighborhood or their field of view (FOV), also in our simulator road users, like cars, detect the events of other dynamic entities in their FOV of one segment. The vehicle behavior can be described as the two-regimen discrete model [6] that has only two possibly states: stopped and advancing. An (event trajectory) example behavior of a car that accelerates, advance at same velocity, break, advance with low velocity, break and finally stay stopped, is shown In Fig. 5. In this figure each point labeled as e_x, indicates an instant where an event is fired and the vehicle changes its state and execute some cognitive procedures (search gap, search obstacle, etc.) which are used to define the next event instant and the state type that will be initialized.

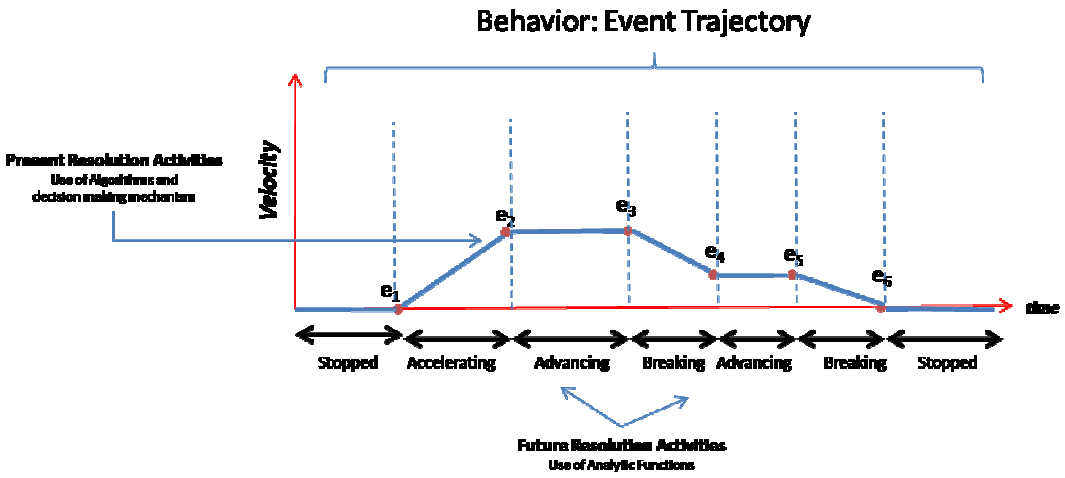


Fig. 5: Behavior definition using the event trajectory

The actor behavior (behavioral data generated by the actor) is determined by two components:

- A function δ_{type} that define the different long term state possibilities that the actor could stay (stopped, accelerating, advancing, breaking) and the events that the actor can generate in order to change the state and modify some attributes (velocity, position, users, etc.) when are executed.
- *The* dmm_i function, i.e., the decision making mechanism (DMM) of actor i is a reasoning process that provides to the actor with the ability to make rational decisions when a choice is present. DMM allows generating plans to reach actor goals using only actor activities in an intelligent way (rational actor) and use the parameters such as: preferred velocity, acceleration rate, etc. Also are used the evolving rules, traffic policies, memory and actor goals to generate an output from the dmm_i function.

4.1.1. δ_{type} function

The vehicle events are defined by the set:

$Z_{vehicle} = \{arrivalLink, arrival, start, leaveLink, stop, alertstop\}$

The vehicle states are defined by the set:

$P_{vehicle} = \{stopped, advancing\}$

The vehicle attributes is represented by the 6-tuple:

$q_{vehiclej} = (p_j, vel_j, mood_j, users_j, w_j, s_j)$

Where:

- $p_j \in P_{vehicle}$
- vel_j, is the actual vehicle velocity
- $mood_j \in \{0,1\}$, are the humor of the vehicle driver
- $users_j \in [1, capacity]$, is the occupants number of the vehicle
- w_j, is the relative position of the vehicle in s_j segment

The set of vehicle states is represented by:

$Q_{vehicle}$= {q_j | q_j is the state of a vehicle j}

The vehicle is represented by the 7-tuple:

$vehicle_j = (size_j, velmax_j, capacity_j, memory_j, plan_j, q_j, q_{0j})$

Where:

- $size_j$ is the length of the vehicle.
- $velmax_j$, is the preferred maximum velocity of vehicle j.
- $capacity_j$, is the maximum number of occupants of the vehicle.
- $memory_j$, temporary memory containing new or derived facts, using IF-THEN rules.
- $plan_j$ = {s_j | $s_j \in S$ }, is an ordered list of segments that the vehicle must visit, conceptually defined as the vehicle objectives.
- q_j, is the state of the vehicle $q_j \in Q_{vehicle}$
- q_{0j}, is the initial state of the vehicle $q_{0j} \in Q_{vehicle}$

Then the set of vehicles is defined by the set:

$Vehicle$ = { $vehicle_j$ | $vehicle_j$ is a vehicle }

Other road users (cyclists, pedestrians, etc.,) can be defined in a similar way.

The transition state for the vehicle is described by:

$\delta_{type} : Q_{vehiclej} \times Z_{vehicle} \rightarrow Q_{vehiclej}$

Where:

- $type \in$ {*"trafficLight","vehicle"*}
- δ_{type} is defined for the vehicle type specific behavior. It receives the state of a $vehicle_j$ and the event. Then the function returns a new state for the $vehicle_j$. Using the PN formalism the δ_{type} function in fig. 5 is described.

Actor activities can be described by a third level net. In Fig. 6 shows the $NET_{3,1}$ describing the vehicle driver states and their possible events. If a $NET_{3,1}$ transition is fired, then a fact is modified. Each transition (agent event) modifies some of the agent facts; for instance the *endChLn* transition modifies the position fact. These events start a DMM cycle. For other dynamical entities in the UTS, the behavior can be also represented by level 3 nets. The definition of the elements of $NET_{3,1}$ is presented in table 3.

Events

Fired Transitions	Modified Facts
endChLn, arrSeg, stop, leavSeg, crash	Position
endChLn, stop, leavSeg	Velocity
arrSeg	Segment
leavSeg	Mood
Stop	Users

Present Resolution Activities

Fired Transitions	activities
t_1, t_2	searchObstacle, gapSearch, conflictResolution

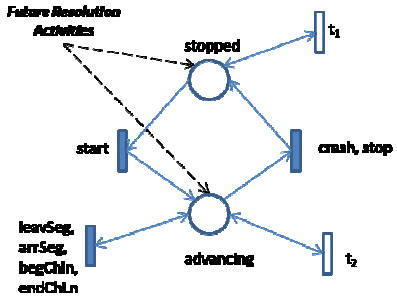


Fig. 6: ***Vehicle Activities Described by $NET_{3,1}$***

Table 1: Table 3: λ,τ and π functions of the $NET_{3,1}$

$NET_{3,1}$
$TOKEN_{3,1} = \{s1\}$; $LABEL_{3,1} = \{endChLn, arrSeg, stop, start, begChLn, leavSeg, crash\}$; $VAR_{3,1} = \{0\}$
λ
$\lambda\{t_3\}=\{(endChLn,^{\uparrow}), (arrSeg,^{\uparrow}), (begChLn,^{\uparrow}), (leavSeg,^{\uparrow})\}$, $\lambda\{t_1\}=\{(start,^{\uparrow})\}$, $\lambda\{t_2\}=\{(stop,^{\uparrow}), (crash,^{\uparrow})\}$
τ
$\tau(p_1) = \tau(p_2) = \{s1\}$
π
$\pi((p_1, t_1), start^{\downarrow})=\pi((t_1, p_2), start^{\downarrow})=\pi((p_2, t_2), stop^{\downarrow})=\pi((t_2, p_1), stop^{\downarrow})=\pi((p_2, t_2), crash^{\downarrow})=\pi((t_2, p_1), crash^{\downarrow})=\pi((p_2, t_3), endChLn^{\uparrow})=\pi((t_3, p_2), endChLn^{\uparrow})=\pi((p_2, t_3), arrSeg^{\uparrow})=\pi((t_3, p_2), arrSeg^{\uparrow})=\pi((p_2, t_3), begChLn^{\uparrow})=\pi((t_3, p_2), begChLn^{\uparrow})=\pi((p_2, t_3), leavSeg^{\uparrow})=\pi((t_3, p_2), leavSeg^{\uparrow})=s1$

Each transition represents an event of the $Z_{vehicle}$ set, and each place a state of the $P_{vehicle}$ set.

4.1.2. Decision Making Mechanism

The dmm_i function, describe the individual DMM for each vehicle. The DMM execution cycle can be described as: Get a plan and execute the plan, specifically the steps are:

- Get a plan. In order to compute a plan the actor goals and the environment information are used. The plan is a sequence of segments to be visited in order to reach the actor goal.
- Decompose the plan into road user activities. In order to decompose the plan into actor activities; first the actor must verify the environment state in its limited FOV, using the shared event list structure and its own state. Afterwards a planning algorithm that takes into account the states, evolving and traffic rules, is executed [20] to translate the plan into an actor sequence of activities.

The dmm_i function for vehicle actor is represented by:

$dmm_{vehiclei}$: $Q_{vehiclei}$ X $Q_{type\backslash FOV} \rightarrow Z_{vehiclei}$

Where:

- *type* ∈ {"*vehicle*","*trafficLight*","*pedestrian*"} are all the possible type of events that can affect the vehicle decision.

- *FOV*, indicate the selected neighborhood states area.

The dmm_i concept and their relation with the $NET_{3,1}$, the environment and the future activities resolution is described in the next figure:

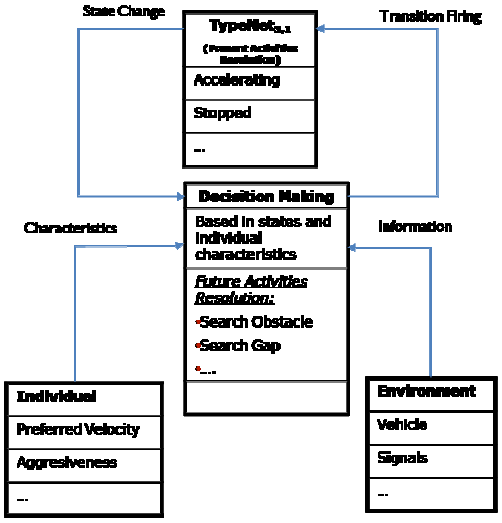
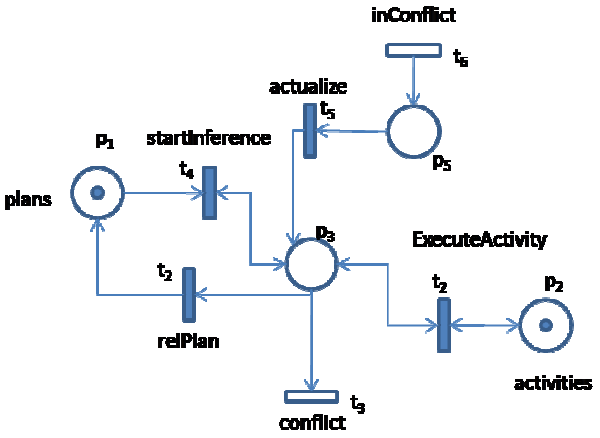


Fig. 7: Decision making mechanism description

The function dmm_i returns a new event based on a reasoning mechanism, taking into account, other actor's states in the FOV of the entity and the actor based knowledge represented by IF-THEN rules. Also are executed the present resolution activities, by example: search for obstacle, search gap, etc., that are executed in the instant of the decision making cycle.

The decision making mechanism (DMM) of an agent is described by the net $MobileNET_{2,1}$ showed in Fig. 8. During the reasoning process, the evolving rules and traffic policies are taken into account. Must be observed that the evolving rules are only an information for the agent, since the execution of this rules are executed in a higher level, this capture the similitude to the real world, by example the gravity law is something that all the humans know about it, but we don't control the execution of this law. In Table 2 the elements of $MobileNET_{2,1}$ are presented.



Fired Transitions	Modified Facts
actualize	Position, velocity, segment, mood, users

Fig. 8: ***Decision Making Mechanism Described by MobileNET$_{2,1}$***

Table 2: Table 2: λ,τ and π functions of the net type *MobileNET$_{2,1}$*

MobileNET$_{2,1}$
TOKEN$_{2,1}$ = {*s1*,*Net$_{3,1}$*}
LABEL$_{2,1}$ = {*conflict, relPlanTask, startInference, actualize, inConflict,endChLn, arrSeg, stop, start, begChLn, leavSeg, crash*};
VAR$_{2,1}$ = {**x** : *typeNet$_{3,1}$* , **y**: *typeNet$_{3,2}$*}
λ
$\lambda\{t_1\}=\{(Stop,^{\downarrow}), (Start,^{\downarrow}), (endChLn,^{\downarrow}), (begChLn,^{\uparrow\downarrow}),(leavSeg,^{\uparrow\downarrow}) , (crash,^{\uparrow\downarrow})\}$, $\lambda\{t_3\}=\lambda\{t_6\}=\{(inConflict,^{=})\}$
τ
$\tau(p_1)=\tau(p_3)=\{s1, typeNet_{3,1}\}$, $\tau(p_2)=\{typeNet_{3,2}\}$,$\tau(p_5)=\tau(p_4)=\{s1\}$
π
$\pi((p_1, t_1), stop^{\downarrow})=\pi((t_1, p_1), stop^{\downarrow})=\pi((p_1, t_1), start^{\downarrow})=\pi((t_1, p_1), start^{\downarrow})=\pi((p_1, t_1), leavSeg^{\downarrow\uparrow})=\pi((t_1, p_1), leavSeg^{\downarrow\uparrow})=s1$
$\pi((p_3, t_3), inConflict^{=})=\pi((t_3, p_3), inConflict^{=}) = \pi((t_6, p_5), inConflict^{=}) =s1$
$\pi((p_2, t_1), stop^{\downarrow})=\pi((t_1, p_2), stop^{\downarrow})=\pi((p_2, t_1), start^{\downarrow})=\pi((t_1, p_2), start^{\downarrow})=\pi((p_2, t_1), leavSeg^{\downarrow\uparrow})=\pi((t_1, p_2), leavSeg^{\downarrow\uparrow})=\mathbf{x}$
$\pi((p_3, t_1), leavSeg^{\downarrow\uparrow})=\pi((t_1, p_3), leavSeg^{\downarrow\uparrow})=\mathbf{y}$

In order to model the traffic lights the on-line traffic flow data to modify the split/cycle/offset timing of the road junction lights is used.

The resource conflict situation generated when an agent tries to execute one action that violate one evolving rule is captured in the model through the transitions *actualize, inConflict,* and *conflict*. It works as follows: the vehicle *i* decides to execute a change lane to lane *k*, however the vehicle *j* is using this lane, then vehicle *i* detects that the 'ubiquity E-Rule' (see section 4.2.4) is violated. However, it decides execute the selected action (here is capture a certain grade of irresponsibility or negligence of the real world). But, before firing the 'negligence E-Rule', is sent a message through the *outConflict* transition to the vehicle *j* in conflict. The vehicle *j* receives the message through the transition *inConflict* then the *actualize* transition fires allowing the vehicle to actualize its state when the vehicle *i* executes the change lane action, and then initialize a reasoning process (*startInference* transition is fired) to made a decision about the new event and maybe stop or even a crash event ('negligence E-Rule').

4.1.3. Decision making cycle

The cognition event vector (CEV) is a set of events that the actor could read or sense from the environment. Each sensed element is described as:

CVE[*i*], where *i* is the *i* sensed element.

Each event has an associated timestamp, and is represented by:

CVD[*i*].time, is the timestamp of the event sensed *i*.

In the reasoning process are used classifiers. Classifiers are a set of rules (IF-THEN) that allow to the agent identify the current situation.

Each time an event is executed the agent compares the CVE (IF side of the classifier) to the list of the existing classifiers. The agent has an inherent tendency to maintain a straight path; if possible, the agent determines the next move (event). Each task module influence in the next event to be executed based on the humor, character, aggressiveness, etc. of the agent. In fig. 5 the DMM cycle is drawn.

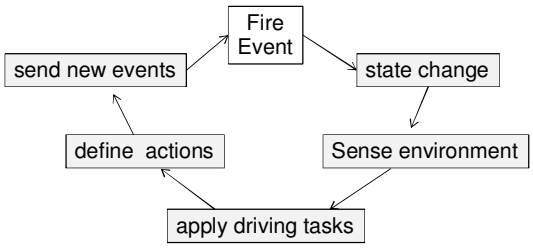


Fig. 9: Decision making cycle

4.1.4. Urban Traffic Policies

There exist some traffic policies that a typical vehicle driver must follow, for instance:

B-Rule 1: Not driving at speeds above the local maximum.
B-Rule 2: Driving on the right side of the road as much as possible.
B-Rule 3: Driving in the allowed direction
B-Rule 4: An entity must travel using the road network

Only aggressive drivers have a tendency to break down some of these rules. This tendency is represented by entity parameters and attributes. These rules are captured as rules in the based knowledge of the actor.

4.1.5. Evolving Rules

Evolving rules are the laws imposed by physical constraints of the modeled system. The following list is the set of selected evolving rules:

E-Rule 1: *Ubiquity rule*. Two entities cannot occupy the same space at the same time.
E-Rule 2: *Ignored entity rule*. An entity cannot pass through another entity.

E-Rule 3: *Negligence rule*. If one agent or more try to violate one or more of the previous rules, then the result is the crashing of the agent and other involved agents.

When an entity tries to execute one action in such a way that one evolving rule could be violated, then it must reconsider the action and select a new action. Otherwise Negligence rule will be fired and the entity will crash.

4.1.6. Methodology capture rules knowledge

To effectively grow a knowledge base in complex operating environments where the relation between state and action spaces is often non-linear, for instance the UTS, are used classifiers. A collection of classifiers represents the rules (knowledge base), which lead the agent know in which situations is now or to define the target [25]. Follow is described the methodology proposed to capture and grow the agent vehicle driver knowledge base.

1) Identify all those possible events (*interacting events*) of the entities (*interacting entities*) the vehicle agent would face during the simulation. This events belong to two kinds of entities:
    a. Vehicle class entity events, it is of the same class entity, for instance: start, stop, etc. This are obtained from the vehicle state driver model (see Fig. 6).
    b. Other entities class events, for instance: a traffic light change, a pedestrian advance, etc.
2) Identify the interacting entity states after their correspondent interacting event is fired. These are obtained from the correspondent entity model. For instance, the vehicle could be in stopped or advancing state (see Fig. 6) or the traffic light could be in *in_green* or *in_red state*.
3) Make a classifier graph for each one of the vehicle driver events. A classifier graph is a Cartesian graph, where the x-axis represents the field of view of the vehicle, for instance, could be numbered from 0 (the beginning of the segment) to the end point of the segment. The y-axis represents the elapsed time and capture the present, future and past.
4) Select one of the possible interacting entities and draw the entity behavior using their events in the classifier graph. The directed line from one event to another is named *state line* and the set of sensed events by the vehicle is called *cognitive event vector*.
5) Select one of the vehicle driver events and put it on the classifier graph before the position and time of the first interacting event of the drawn entity behavior.
6) Draw all the possible *rule lines* in the classifier graph. A rule line is a directed line from the vehicle driver event to the intersection of an interacting event state line before and after it.
7) Generate the IF-THEN rules (classifiers) that govern the movement of the agent. Each classifier is composed of two parts: the IF part or the cognition event vector presents an image of the agent´s surroundings. The THEN part (action) is the event to be executed to change the vehicle state.

a) Vehicle-traffic light case study

The methodology implementation is introduced through the vehicle-traffic light case study. In Fig. 10 the correspondent case study is shown. A segment with an arrival car event at the beginning of the segment and a traffic light at the end of the segment with a light change event (LCE) are drawn.

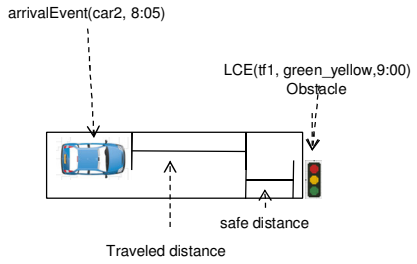


Fig. 10: Classifier vehicle-traffic light graph

In the Fig. 11 is presented the classifier graph of the study case. The obstacle presented is the traffic light at the position 55 at the end of the segment, and the correspondent state and rule lines.

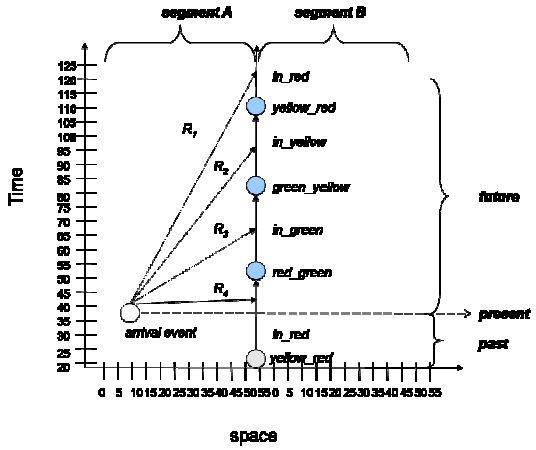


Fig. 11: Classifier vehicle-traffic light graph

For instance, from instant 55 to 85 the traffic light will be on green light state. The event of the vehicle evaluated is the arrival event, it means, when a new vehicle arrives to the segment from another segment arrives at position 0 of the segment at instant 35.

The rules lines generated are R1 to R4. R1 describes the situation of the vehicle arriving to a segment, and the calculated VAT is greater than the last light change. Then the vehicle will generate a new stop event.

As stated by [26] the driving cannot be reduced to rule-based behavior. Because the navigation and especially the control level of car driving are seldom rule based, but rely heavily on knowledge or sills, then as can be observed the THEN clause obtain an individual value of the vehicle agent, for instance the *delayToStart* or *safeDistance* parameters. The obtained rules for a vehicle agent using the proposed methodology are presented in Table 3.

Table 3: IF-THEN rules acquired using the proposed methodology for the vehicle-vehicle case study

Rule Line	Cognitive Event Vector (CEV)	Action

R_4	IF CEV[i] == LCE_GREEN and VAT < CEV[i].time	THEN vehicle.new(SE.time = VAT) and vehicle.new(BE.time = CEV[i].time + vehicle.getDelayToStart)
R_2	IF CEV[i] == LCE_RED and VAT < CEV[i].time	THEN vehicle.new(CE.time = VAT)
R_3	IF CEV[i] == LCE_YELLOW and VAT < CEV[i].time	THEN vehicle.new(CE.time = VAT)
R3	IF CEV[i] == LCE_GREEN and VAT >= CEV[i].time	THEN vehicle.new(CE.time = VAT)
R1	IF CEV[i] == LCE_RED and VAT >= CEV[i].time	THEN vehicle.new(SE.time = VAT) and vehicle.new(BE.time = CEV[i].time + vehicle.getDelayToStart)
R2	IF CEV[i] == LCE_YELLOW and VAT >= CEV[i].time	THEN vehicle.new(CE.time = VAT)

b) Vehicle-vehicle case study

The traffic light is not the unique possible obstacle that a vehicle could found. In fig. 8 is shown the case when a vehicle have another vehicle in front of it.

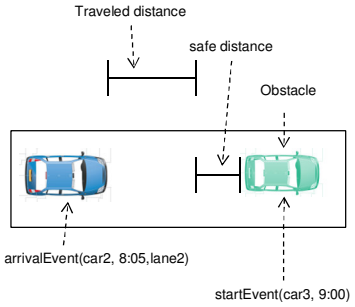


Fig. 12: Vehicle-vehicle case study

In this case all the possible events that can form a CEV could be obtained from the PN representation of fig. 2, a vehicle can be stopped, after a time can start and then leave the street.

In the Fig. 13 is presented the classifier graph of the study case. The obstacle presented is the vehicle at the position 26 of the segment, and the correspondent state and rule lines.

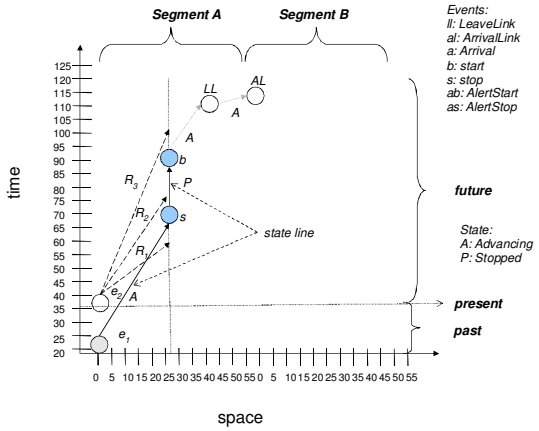


Fig. 13: Vehicle-vehicle classifier graph using a vehicle-reference system consisting of a representation of the time and space axes and the cognition vector event.

In table 1 is shown the rules captured for the vehicle agent using the proposed methodology.

Table 4: IF-THEN rules captured from the methodology proposed for the vehicle-vehicle case study.

Rule Line	Cognitive Event Vector (CVE)	Action
R_3	IF CEV[i] == BE and VAT >= CEV[i].time	THEN vehicle.new(WSE.time = VAT)
	IF CEV[i] == WBE and VAT >= CEV[i].time	THEN vehicle.new(WSE.time = VAT)
	IF CEV[i] == CE and VAT >= CEV[i].time	THEN vehicle.new(WSE.time = VAT)
R_2	IF CEV[i] == BE and VAT < CEV[i].time	THEN vehicle.new(SE.time = CEV[i].time+ vehicle.getDelayToStart)
	IF CEV[i] == WBE and VAT < CEV[i].time	THEN vehicle.new(WSE.time = CEV[i].time)
	IF CEV[i] == CE and VAT < CEV[i].time	THEN vehicle.new(WSE.time = CEV[i].time)

4.2. Environment Model

The environment model is those system resources that are available to the system users. The resources can be needed by users to achieve their goals, or can be used to provide information to users. Also is used by stimulus subsystems (for instance, UTCM, demographic information system, among others) to implement their policies. Thus, the entities of environment model are: road network and the traffic information static and dynamical signals

4.2.1. Traffic Information Static Signals

The static traffic signs are represented as objects. Each object is represented by the 2-tuple:

$o_s = (type_s, w_s)$

Where:

- $type_s$ = *(signal, descriptor),* and *signal* could be any UTS traffic signal (speed-limit, one way, turn to left, bus stop, bumps, etc.) and *descriptor* is a supplementary static signal information, for instance 80 km/hr for the speed-limit signal.
- $w_s \in [0, gps\_location]$, is the object position in the road with respect to a geographical positioning system location.

4.2.2. Traffic Information Dynamic Signals

The dynamic traffic signs are represented also as objects, but with the difference that the descriptor value is variable and could be modified by a stimulus subsystem, and are represented by the 3-tuple:

$$o_d = (type_d, w_d, \beta_{type})$$

Where:

- $type_d$ = *(*$signal_d$*,* $descriptor_d$*),* and $signal_d$ could be any UTS traffic signal (variable message sign (VMS), traffic light, etc.) and $descriptor_d$ is the dynamic signal information value, for instance red or green for a traffic light.
- $w_d \in [0, gps\_location]$, is the object position in the road with respect to a geographical positioning system location.
- β_{type} function that describes the behavior of this kind of objects. Since this kind of entities doesn't have any kind of decision making as the actors have, only are described the possible states and events.

4.2.3. β_{type} function

As stated earlier the entity behavior, (see section 4.1) is described as an event trajectory. Then the traffic light behavior can be described by a trajectory of change color events, for instance: green to yellow, yellow to red, etc. The traffic light events are described by:

$$Z_{trafficLight} = \{red\_green, green\_yellow, yellow\_red\}$$

The possible traffic light states are represented by the set:

$$Q_{trafficLight} = \{in\_green, in\_red, in\_yellow\}$$

The traffic light state is represented by the 3-tuple:

The other dynamic traffic signal, variable message signs, guidance dynamic route, parking guidance, etc., would be defined in the same way.
Then transition state function for the traffic light is described by:

$$\beta_{type}: Q_{trafficLightk} \text{ X } Z_{trafficLightk} \rightarrow Q_{trafficLightk}$$

Where:

β_{type} is defined for the traffic light type specific behavior. It receives the state of a $trafficLight_k$ and an event for this traffic light. Then the function returns a new state for the $trafficLight_k$. Using the $NET_{3,2}$ is described the β_{type} function. See Fig. 14.

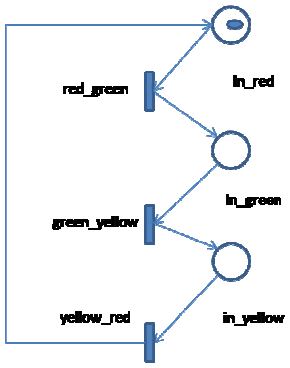


Fig. 14: $NET_{3,2}$ for traffic light behavior

The events of the $NET_{3,2}$ traffic light actors can be controlled by a stimulus subsystem, for instance an intelligent transportation system (ITS).

4.2.4. Road network

The road network is a set of interconnected street lanes and intersections; its main function is to contain the dynamic and static traffic information signals and road users which transverse through the network.

In the model the road network is represented by segments and two relationships to interconnect segments. In order to obtain each segment, streets must be divided by continue lanes, as show in Fig. 15. Each segment s_i is represented using the 4-tuple $s_i = (a_i, b_i, O_i, type_i)$, where:

- $a_i, b_i = (lat_i, long_i)$ and $lat_i, long_i \in \Re$, which are the latitude and longitude of the two segment end points, and could exist an entities movement from a to b, or vice versa.
- O_i is the set of objects in the segment i, $O_i \subseteq O$
- $type_i \in \{street, crosswalk, intersection\}$, represents the type of the segment.

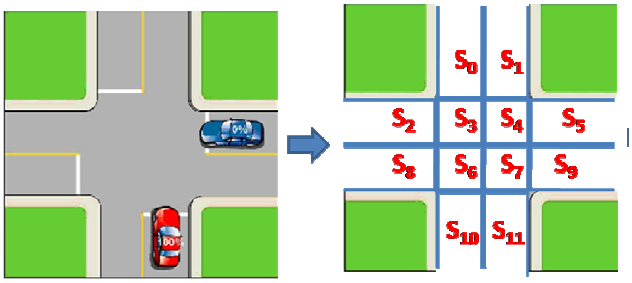


Fig. 15: Real intersection and intersection described using segment components

The segments that are located in the borders of the traffic network are named source and sink nodes; they are considered as inputs and outputs of vehicles to the road network.

Before describing the methodology to capture the connection between segments the next relations are stated:

Relation 1: Contiguous neighborhood, $NC = \{ (t_{ij}, p_i, p_j) \mid t_{ij} \in T, p_i, p_j \in P$, the entities that flow throw t_{ij} could be inserted into any position of segment described by $p_j \}$

Relation 2: Sequential neighborhood, $NS = \{ (t_{ij},p_i,p_j) \mid t_{ij} \in T, p_i,p_j \in P$, the entities that flow throw t_{ij} could be added to tail position of segment described by p_j }

In order to capture the connection between segments, the following methodology is proposed. Using this strategy the model $EnvironmentNet_1$ is obtained; the initial marking will be defined in subsequent paragraphs. This procedure is shown in Fig. 16.

1. For each segment $s_i \in S$, *a place* p_i is assigned.
2. A transition t_{ij} is added for every (s_i,s_j) physically related (that exists a physical connection), together with arcs (p_i, t_{ij}) and (t_{ij},p_j) and (p_j, t_{ji}) and (t_{ji},p_i) and define kind of relation, contiguous or sequential.

Furthermore, some transitions t_i must be added for every segment s_i source or sink; arcs (t_i,p_i) or (pi,ti) are added accordingly.

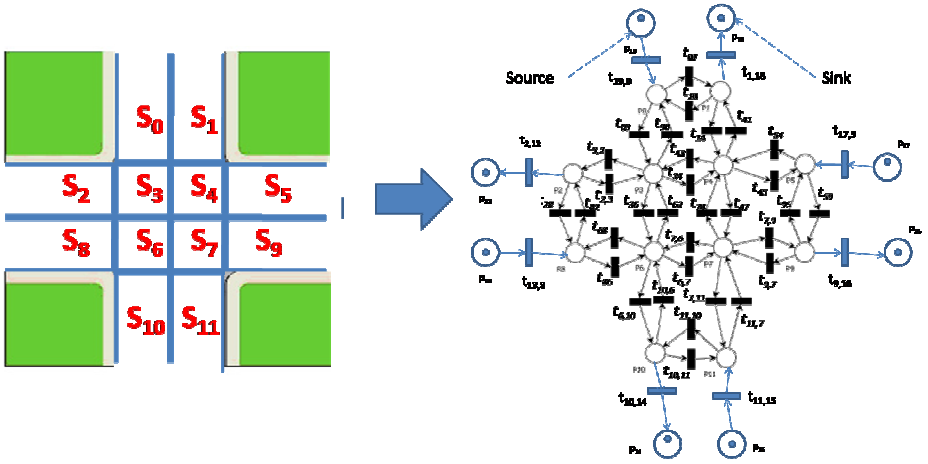


Fig. 16: $EnvironmentNet_1$

Intersection Segment Definition: a segment that has one or more inputs-outputs in addition of the two segment end points (see segment definition).

As could be observed, the in Fig. 15, there are four intersection segments. To obtain a reduced definition of intersection, the next methodology is followed (see Fig. 17):

1. For each place $p_i \in P$, *that compose an intersection, select the max subset of places and replace by one new place* p_n. This set must be composed of places of the same lane.
2. All the previous transition t_{ij} must be directed to the place p_n that replaces the old places.
3. Then assign each transition t_{ij} , t_{ji} of the p_i deleted place a relation *NC* or *NS* as correspond in the next way:
    1. All the arcs that have the same side input in the new place p_n , then all belong to the NC relation.
    2. All arcs that are the extreme places of the new place are added as *NS* relations.
    3. And all the new places interconnected by the new transitions, belong to the *NC* relation.
    4. If in the original segment representation, there exists only one segment, then is selected in an arbitrary way the *NC* or *NS* relation.

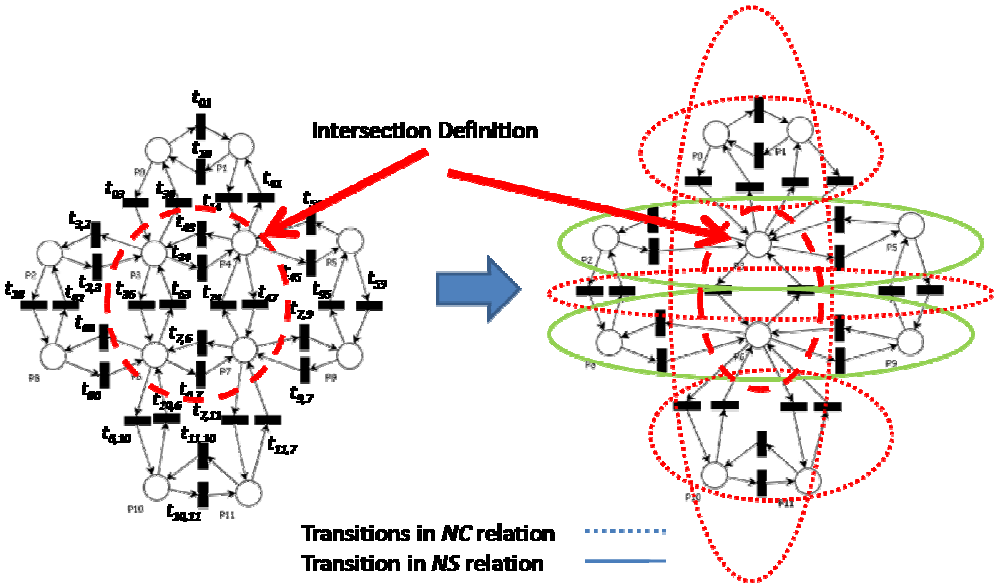


Fig. 17: Intersection definition using segment components.

Once defined the *NC* and *NS* relations then are assigned to each transition the correspondent label that describe the movement from one place to another. To all the transitions contained in the *NC* relation a label "*begChLn*" is assigned, and for all transitions contained in *NS* relation, a "*leavSeg*" label is assigned. The static traffic signals for instance speed limit, bumps position, segment size, are information sent to the actor when a *leaveSeg* transition is fired (t_{06}, t_{17}, etc.). The tokens contained in the net, are described by the $AgentNET_2$ types, for instance $MobileNET_{2,1}$ that represents the vehicle actors of the system. In Table 5 are summarized the components of the environment model.

Table 5: λ,τ and π functions of the net type $EnvironmentNET_1$

$EnvironmentNET_1$
$TOKEN_1$ = { $MobileNET_{2,1}$}
$LABEL_1$ = {*leavSeg, begChLn*}
$VAR_{1,1}$ = {**x** : $MobileNET_{2,1}$ }
λ
$\lambda\{t_{01}\}=\lambda\{t_{10}\}=\lambda\{t_{67}\}=\lambda\{t_{76}\}=\lambda\{t_{34}\}=\lambda\{t_{43}\}=\{(\text{begChLn},^{\downarrow})\}$,$\lambda\{t_{06}\}=\lambda\{t_{17}\}=\lambda\{t_{62}\}=\lambda\{t_{57}\}=\lambda\{t_{63}\}=\lambda\{t_{74}\}=\{(\text{leavSeg},^{\downarrow})\}$
τ
$\tau(p_1) = \tau(p_2) =...= \tau(p_n) = \{MobileNET_{2,1}\}$ *n* is the number of segments
π
$\pi((p_0,t_{01}),\text{begChLn}^{\downarrow})= \pi((p_1,t_{10}),\text{begChLn}^{\downarrow})= \pi((p_6,t_{67}),\text{begChLn}^{\downarrow})= \pi((p_7,t_{76}),\text{begChLn}^{\downarrow})= \pi((p_3,t_{34}),\text{begChLn}^{\downarrow})=$ $\pi((p_4,t_{43}),\text{begChLn}^{\downarrow})=$ $\pi((p_0,t_{06}),\text{leavSeg}^{\downarrow})= \pi((p_1,t_{17}),\text{leavSeg}^{\downarrow})= \pi((p_6,t_{63}),\text{leavSeg}^{\downarrow})= \pi((p_7,t_{74}),\text{leavSeg}^{\downarrow})= \pi((p_6,t_{62}),\text{leavSeg}^{\downarrow})=$ $\pi((p_5,t_{57}),\text{leavSeg}^{\downarrow})=$ **x**

4.3. Urban traffic system model

Formally, the UTS can be described as follows:

Definition. A UTS is a 3-tuple, defined by:

STU = (3-LNS, $\{S_i\}$,I)

Where:

- 3-LNS, is the definition of environment and actors models described by the n-LNS formalism in a three level way.
- $\{S_i\}$, represents the set of segments in the UTS.
- *I*, are those $ActivitiesNET_3$ ($NET_{3,2}$,$NET_{3,3}$, etc.) that are description for entities of the stimulus subsystems, i.e. the input events.

The resulting model for correctness and completeness can be checked. The next issues could be checked, beside is stated the justification:

- Check that at least one segment exists: by PN definition.
- Check for no more than one intersection exists at the same point, by the proposed methodology to obtain the intersections.
- Check for each mobile actor exist a synchronized label with an environment label.

Thus, if no isolated segments or crossings, then the description is complete. If all the previous conditions are successfully filled, then the description is correct.

4.4. Implementation issues

4.4.1. Resources Structures

In the proposed model a set of resource allocation tuple that represent the future action execution is defined by:

$r = (a,k,z,p,w)$

Where:

- $a \in A$, is the actor that will execute the event description
- $k \in \Re^+$, is the event execution time
- $z \in Z$ is the event description
- $p \in P$ is the event execution place (segment)
- w $\Re^+$ is the event execution position

Each tuple *r* is a schedule description, which requires a reference to an entity (a) that will execute an event (*z*) in position (*w*) of a place (p) at the instant (*k*).

Then the set of all the resource allocations is described by:

$$R = \{(a, \kappa, z, p, \omega)\}$$

The tuples of the set *R* are related by $\geq$ and Γ relations. In order to describe these relations the next semantics are defined:

- $r_i.p = p \in P$, $r_i \in R$, gets the place (segment) where the event will occur.
- $r_i.k = \tau \in \Re^+$, $r_i \in R$, gets the time at event will occur.
- $r_i.a = a \in A$, $r_i \in R$, gets the actor that will execute the event description.

The next dependency relation Γ represents the relation between $actor_j$ that is observed by $actor_i$, i.e., the $actor_j$ is an obstacle for the $actor_i$:

Relation 1: Dependency:
$\Gamma \subseteq R \text{ X } R = \{ (r_i, r_j) \mid r_i.p = r_j.p$ and $r_i.a$ was observed by $r_j.a$ to make a decision) $\}$

The next relation allows to represent the causality relation of the resource allocations, i.e., r_i is executed before r_j and occurs at the same place (segment):

Relation 2: Causality:

$\leq \subseteq R \text{ X } R = \{ (r_i, r_j) \mid r_i.k \leq r_j.k$ and $r_i.p = r_j.p \ \}$

In Fig. 18, is represented the set R with their correspondent relations.

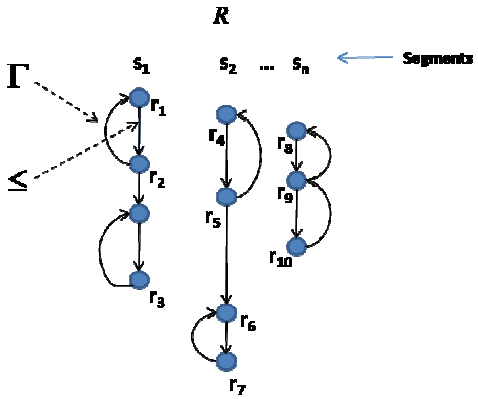


Fig. 18. Space-Time Structure of Events Trajectory

Next are defined the axiom that allows to capture that situation of if a place p_i contains actors, then there must be at least one event for each actor in the place p_i to be executed.

Axiom 1: Actors Existence

$\forall x (x \in A \Rightarrow \exists r \in R,\ r.a = x)$

The next axiom are added to capture de evolving rules described for UTS, by example the axiom 2 describes the physical constraint that a vehicle cannot pass over the in front vehicle (evolving rules implementation):

Added-Axiom 1: Sequentiality

If $\exists x,y,z \in R$ and $(x,y) \in \Gamma$ and $(x,z) \in \Gamma \Rightarrow y = z \vee x.a \in O_s$

$x.a$, refers to the actor of the correspondent event tuple, this part of the axiom, allow that actors as the traffic lights (O_s) can be observed by other entities in the same segment.

Thus can be defined a global function T, that executes only the minor of all the events in all the segments. The execution of this function is done for a given simulated time, and is described in the next paragraph:

$Ne = \{(a,k,z,s,w) \mid (a,k,z,s,w) \in E \wedge k = \min\{eventList_i\}_{i=0}^{n}\}$
$n = \#\{S_i\}$
$a = Ne.a$
$\delta_{type}(a.q_{type}, Ne)$

4.4.2. Stimulus Population generation subsystem

In order to create new entities as the road users, a list of source segments are specified in the population generation system. The model used in this research is a Poisson distribution, but could be substituted by real demographic data. The Poisson distribution is noted by the formula:

$$P(n)=((\lambda t)^n e^{-\lambda t}) / n!$$

Where:

$P(n)$ is the probability of exactly n vehicles arrives at time t.
λ is the average arrival rate (*veh/ min*).
t is the duration of time over which the vehicles are counted.

Each activity consists of a type and priority and defining the start and ending time currently is used a set of static activities but can be expanded to use more complex models (i.e. an OD matrix), is used only cars but can be added easily a subsystem to specify a transportation mode for each activity depending on certain values, which could be economical status, distance, comfort, etc.

4.4.3. Stimulus Traffic Management Control subsystem

All traffic control systems currently used, often operate on the basis of adaptation to the times of the green phases. These methods are not optimal even when the traffic demands changing rapidly in a time interval. Could be implemented as a predefined green phase's time, or in accord with some time lines change the green phase time. Although, could be implemented in a more complex way, with adaptive traffic control. This subsystem generates the events of the traffic lights described in section 4.2.3.

4.4.4. Case study: City section

A simulation with a network structure of 38 segments (Fig. 19), with 1000 vehicles, 60 traffic lights was performed. Each intersection has four traffic lights and each street have only one lane flow. The demographic info proposed in section 4.4.2 and the UTCM proposed in section 4.4.3 are used as stimulus subsystems to the simulator.

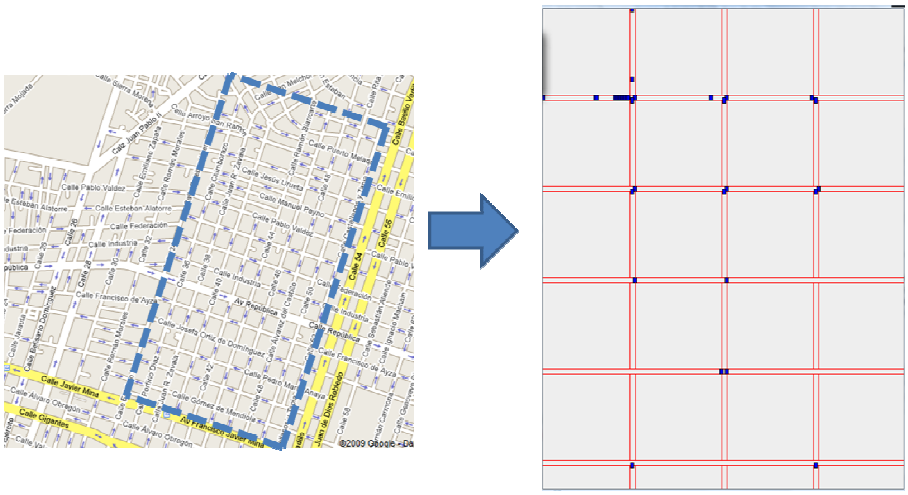


Fig. 19: a) real UTS b) UTS simulator

The Fig. 20 depicts the flow-density relation for a specific road. The *x* axis shows the density and *y* axis the flow. Observe how the flow-density fundamental diagram is conserved, it means that, if the density increases it value the flow decreases its value, as observed in the circle area.

The "strategy change" line indicates the instant at which the traffic light changes their control strategy. When is used the *A* strategy, the red time is set to 60 seconds and the green time to 20 seconds and when the *B* strategy is selected, the red time is set to 20 seconds and the green time to 60 seconds. The registered flow when *B* Strategy is used results to be higher than when the *A* strategy is used.

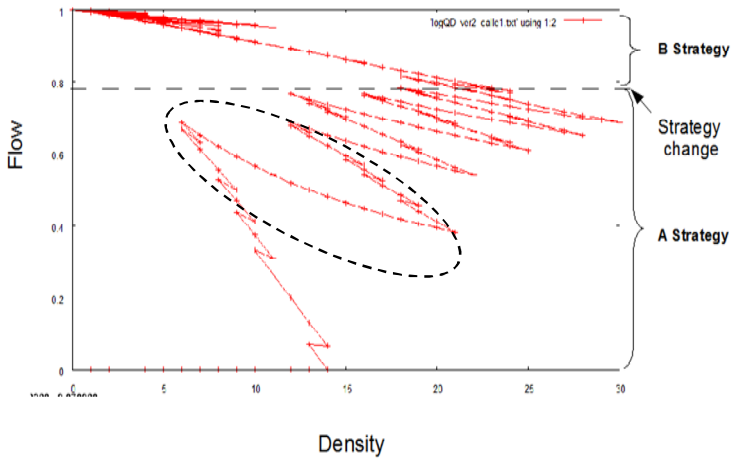


Fig. 20: Flow-Density relationship.

5. Conclusions

In this paper a modeling framework for urban traffic systems using relationships and sets to capture the urban traffic network and some geographical information was presented. Also in the context of multi-agent systems, the presented work contributes to the development of a more general model to capture the dynamic of urban traffic systems (vehicle, traffic light, pedestrians, etc).

The proposed model allows to select the microscopic desired level of road user behavior information. Also it is included the field of view concept which allows to reduce the obstacle area search by a road user entity. As a final remark, external models interaction is introduced through the DMM.

REFERENCES

[1] M. Chabrol, D. Sarramia, and N. Tchernev, "Urban Traffic systems modeling methodology", *International Journal of Production Economics*, vol. 99, Feb. 2006, pp. 156-176.

[2] W. Chaker, B. Moulin, and M. Thériault, "Modeling Transport networks with Design Pattern: Application to Hybrid Traffic Simulations*", in Proc. 18th IASTED International Conference Modeling and Simulation*, May. 2007, pp. 568-573.

[3] A. Di Febbraro, D. Giglio, and N. Sacco, "Modular representation of urban traffic systems based on hybrid Petri nets," in *Proc. IEEE Int. Conf. Intelligent Transportation Systems*, Aug. 2001, pp. 866–871.

[4] A. Drogoul, D. Vanbergue and T. Meurisse. (2002). "Multi-Agent Based Simulation: Where are the agents?" *in Proc. Of Multi-Agent Based Simulation*, Jul. 2002, pp. 1-15.

[5] J. K. Lee, Y. H. Lim, and S. D. Chi, "Hierarchical modeling and simulation environment for intelligent transportation systems," *SIMULATION*, vol. 80, no. 2, Feb. 2004, pp. 61-76.

[6] E. López-Neri, E. López-Mellado, and A. Ramírez-Treviño, "Mobil Agent Systems based urban traffic micro-simulation," M.S. thesis, Dept. Electron. Eng., CINVESTAV, Guadalajara, Jalisco, México, 2005.

[7] B. D. Lubachevsky, "Relaxation for Massively Parallel Discrete Event Simulation," *in Performance Evaluation of computer and Communication Systems, Joint tutorial Papers of Performance '93 and Sigmetrics '93*, L. Donatiello and R. D. Nelson, Eds. Lecture Notes in Computer Science, vol. 729, Springer-Verlag, London, pp. 307-329.

[8] K. Nagel, "Traffic Networks," *in Handbook of Graphs and Networks*, S. Bornholdt and H.G. Schuster, Eds., Wiley-VCH, 1992, pp. 248-272.

[9] P. Wagner, and M. Wegener, "Urban Land use, Transport and Environment Models: Experiences with an Integrated Microscopic Approach," *in DISP,* vol. 170, March 2000, pp. 45-46.

[10] L. Correia, and T. Wehrle, "Beyond cellular automata, towards more realistic traffic simulators," *in Proc. Of Cellular Automata - 7th International Conference on Cellular Automata for Research and Industry ACRI 2006*, Perpignan, France, September 2006, vol. 4173, Springer-LNCS, pp. 690-693.

[11] J. W. Barclay, "*Point-to-Point microscopic simulation: a discussion of issues,*" in Proc. of the 1st Western Pacific and 3rd Australian Japan Workshop on Stochastic Models in Engineering, Technology and Management, Christchurch, September1999, pp.176-81.

[12] G. Wainer, N. Giambasi, "Specification, modeling and simulation of timed Cell-DEVS models". Comp. Dpto. FCEN/UBA, Tech. Rep. 97-007, Nov. 1998.

[13] H. V. D. Parunak, T. Belding, S. Brueckner and J. Sauter, "Emergent Behavior Modeling, " *in Advanced Campaign Planning*, Alexander Kott and Steve Morse, Eds., submitted for publication.

[14] P. A. M. Ehlert, and L. J. M. Rothkrantz, "Microscopic traffic simulation with reactive driving agents," in Proc. of the IEEE Intelligent Transportation Systems Conference, August 2001, pp. 860-865.

[15] J. Dijkstra, A. J. Jessurun, B. de-Vries, and H. J. P. Timmermans, "Agent Architecture for Simulating Pedestrians in the Built Environment," *In Proc. of the Fourth International Workshop on Agents in Traffic and Transportation (ATT2006)*, New York, NY, 2006, pp. 8-16.

[16] X. Jin, M. Itmi, and H. Abdulrab, " A cooperative multi-agent system simulation model for urban traffic intelligent control," in *Proceedings of the 2007 Summer Computer Simulation Conference* (San Diego, California, July 16 - 19, 2007). Summer Computer Simulation Conference. Society for Computer Simulation International, San Diego, CA, pp. 953-958.

[17] A. Cicortas, and N. Somosi, "Multi-Agent System Model for Urban Traffic Simulation," in SACI '05: 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, May 12-14, 2005.

[18] D. Weyns, A. Omicini, and J. Odell, "Environment as a first class abstraction in multiagent systems," in *Autonomous Agents and Multi-Agent Systems,* vol. 14, no. 1, pp. 5-30, Feb. 2007.

[19] B. P. Zeigler, *Theory of Modeling and Simulation*, Wiley, New York, 1976, pp. 50-69.

[20] Beckman, R. J., Baggerly, K. A., & McKay, M. D. (1996). Creating synthetic base-line populations. Transportation Research Part A - Policy and Practice , 30(6):415-429.

[21] K.M. Vaughn, P. Speckman, and E.I. Pas. Generating household activity-travel patterns (HATPs) for synthetic populations, 1997.

[22] J. L. Bowman. The day activity schedule approach to travel demand analysis. PhD thesis, Massachusetts Institute of Technology, Boston, MA, 1998.

[23] R. Sánchez-Herrera, and E. López-Mellado, "Modular and Hierarchical Modeling of Interactive Mobile Agents," in IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, 10-13. pp. 1740-1745 vol.2. (2004).

[24] R. Valk, "Petri Nets as Token Objets. An introduction to Elementary Object Nets," in Int. Conf. on application and theory of Petri nets, pp:1-25, Springer-Verlag (1998)

[25] D. Goldberg, "Genetic Algorithms", Addison-Wesley, Reading, MA, 1989.

[26] H. Dreyfus, and S. Dreyfus, "Mind over Machine: the power of human intuition and expertise in the era of the computer", Oxford; Basil Blackwell, 1986.
[27] E. López-Neri, E. López-Mellado and A. Ramírez-Treviño, "Microscopic Modeling Framework for Urban Traffic Systems Simulation", in the 7th International conference on system simulation and scientific computing, 2008. Beijing, China.