Modeling and Formal Analysis of Urban Road Traffic

Camelia Avram[a], José Machado[b] and Adina Aştilean[a]

[a]*Department of Automation, Automation and Computer Science Faculty, TUCN, Cluj Napoca, Romania*
[b]*CT2M Research Centre, Mechanical Engineering Department, University of Minho, Guimarães, Portugal*

Abstract. Modern life in cities leads to complex urban traffic road and, sometimes, to go from one point to another, in a city, is a hard and very complex task. The use of assisted systems for helping drivers on their task of reaching the desired destination is being common, mainly systems like GPS location systems or other similar systems. The main gap of those systems is that they are not able to assist drivers when some unexpected changes occur, like accidents, or another unexpected situations. In this context, it would be desirable to have a dynamic system to inform the drivers, about everything that is happening "online". This work is inserted in this context and the work presented here is one part of a bigger project that has, as main goal, to be a dynamic system for assisting drivers under hard conditions of urban road traffic. In this paper is modeled, and formally analyzed, the intersection of four street segments, in order to take some considerations about this subject. This paper presents the model of the considered system, using timed automata formalism. The validation and verification of the road traffic model it is realized using UPPAAL model-checker.

Keywords: road traffic modeling; routing algorithms; formal methods; analysis techniques.
PACS: 02.30.Yy

INTRODUCTION

Dependable specification of complex behaviors of discrete and/or real-time discrete events systems can be improved by using several analysis techniques like, for instance Simulation [1], Diagnosis [2], Test [3] and Formal Verification [4], among others. Dependable centralized controllers [5] and/or dependable distributed controllers [6] can be developed using those techniques.

Concerning embedded controllers applied to transportation systems, there are some approaches that consider, together, Simulation and Formal Verification, as complementary techniques [7]: Simulation for achieving, very fast, some important results and, then, Formal Verification for achieving results that cannot be guaranteed by using, only, Simulation.

In this paper it is intended to develop a dependable distributed controller that will be used– not on the transportation systems itself– but as support decision system, for drivers, when they are under extreme conditions of urban road traffic having different options of roads for achievement the final destination. In fact, a dynamic decision support system for drivers – used in this context – has a lot of direct and indirect benefits for users like, for instance: reduction of stress, reduction of time for achievement of final destinations, reduction of emission of pollution; reduction of consumption of energy, among others.

In order to achieve the proposed goal, the paper is organized as follows: section 2 is devoted to the presentation of choices and work hypothesis considered; section 3 presents the results concerning simulation and formal verification tasks; and, finally, section 4 presents some conclusions and main ideas for future work.

HYPOTHESIS CONSIDERED

In this paper the analysis techniques chosen for improvement of distributed controllers' dependability were simulation and formal verification due the same reasons presented in [4]. Concerning Formal Verification technique, there are some approaches that can be used [8], but we decided to use formal verification by model-checking [9] because it is an automatic technique and with good results for users. Due the characteristics of the kind of systems that we intend to model (need of time modeling) there is need to define the formalism and software tools for performing Simulation and Formal Verification tasks.

A number of formalisms can be used to model timed systems. Timed automata [10] were adopted as the modeling formalism for modeling due to two main reasons: first, the study of the proposed system needs to take time into account; and, second, it is the input formalism of the UPPAAL model-checker. Hence, it is well adapted for Simulation and Formal Verification of timed systems. Also, there is the advantage of using one single software

11th International Conference of Numerical Analysis and Applied Mathematics 2013
AIP Conf. Proc. 1558, 1408-1411 (2013); doi: 10.1063/1.4825779

tool for performing Simulation and Formal verification tasks using the same developed models in timed automata. UPPAAL is a toolbox designed to verify systems that can be modeled as networks of automaton extended with integer variables, structured data types, defining functions, and channel synchronization [11].

The Automata Model

Street segments interact via the inflow and outflow of vehicles at their boundary. The only major difficulty is to determine the optimal number of cars passing the observed street to obtain a medium passing speed close to the legal limits (outputting a stream of numbers describing the flow of vehicles in vehicles/minute crossing a boundary). Several situations which can occur are also taken into consideration and are introduced in the simulation scenarios.

The Automaton model for one road segment is presented in FIGURE 1 and for one car is presented in **FIGURE 2**.

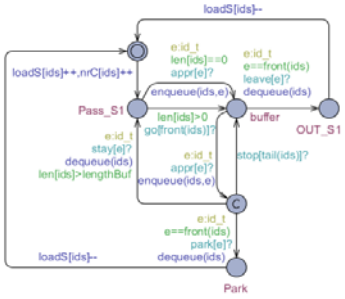

FIGURE 1. The automaton model of one road segment

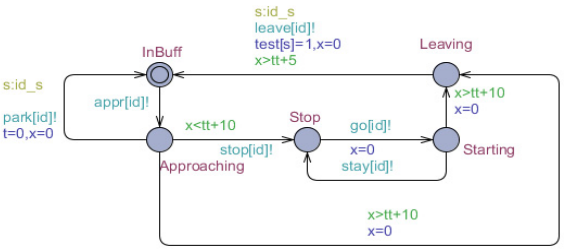

FIGURE 2. The automaton model of one car

A buffer was used to model the number of cells (street size). The synchronization is realized with channels (appr(), go(), stop(), stay() and leave()). Different situations are modeled trough messages exchange (stop and go, parking). Each street segment will have different configuration parameters (length represented by the buffer length, minimal and the maximal time to pass the segment). For each vehicle a "car" object will be instantiated. Based on the driver and vehicle capacity each "car" object will have different values for the "time" variable for braking and accelerating.

An intersection is modeled as a collection of segments, presented in **FIGURE 3**. An exit function should be defined to setup specific trough out rates for each segment. This function is used also in the routing process. In this model the trough out rate for one segment is analyzed, the same model is used for each other segment of the crossroad, varying the configuration parameters at the beginning of the simulation.

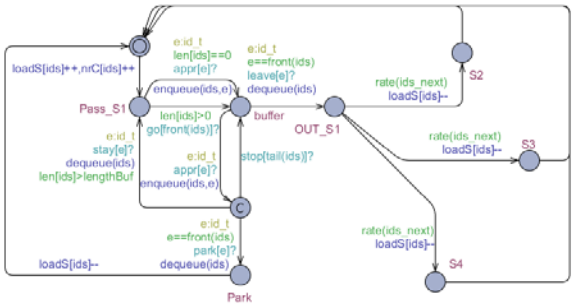

FIGURE 3. Intersection of four streets model

This model is composed by a network of five automata: one automaton to model the vehicle, one automaton to model the street segment, one to model an intersection and two automata to monitor and to collect the values of the variables which correspond to monitored road traffic parameters. For each road segment an automaton is instantiated. To realize a modular and in the same time complex and faithful model to the real life road traffic a solution based on instantiations was chosen; this way the analyzed road traffic map can be changed fast and the model will not suffer any modifications.

To model and simulate a 2 km^2 urban area, 13^{th} intersection, 50 road segments and 200 vehicles the system will have 265 automata (200 instantiations for car autom., 50 instantiations of the street autom., 13 instantiations of the intersection autom. and one instantiation of the two autom. related to observe the traffic behavior). Trough out rates

for one analyzed intersection are: Segment 0/Segment 1:0.30; Segment 0/Segment 2:0.50; Segment 0/Segment 3:0.20.

The variation of the vehicle number versus street segment buffer load during simulation time units is presented in **FIGURE 4**. A color representation is use to emphasize the traffic type.

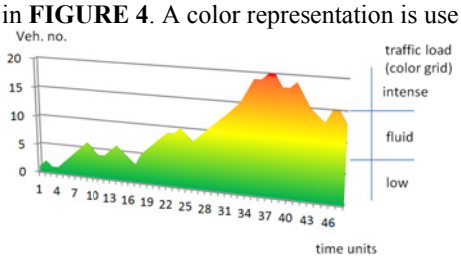

FIGURE 4. Variation of the buffer load versus intensity

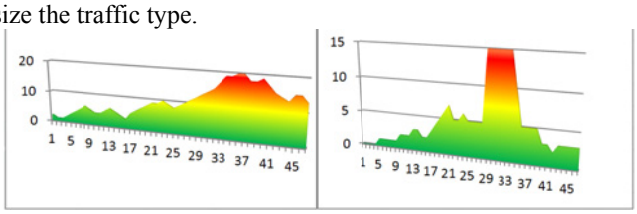

FIGURE 5. Congestion propagation between two segments

The representation of two segments connected is presented in **FIGURE 5**. *Segment 1* has a capacity equal to 20 and *Segment 2* is smaller, it has a buffer of 15 positions. The exit of the first road segment is the entrance of the second road segment. Simulating a deadlock (from various reasons) on the upper segment will determine the propagation of the congestion. Starting with the 25th until 37th time units *Segment 2* is blocked and we can observe that the *Segment 1* is starting to get congested. During simulation this scenario was implemented as an event connected to a specific time interval.

SIMULATION AND FORMAL VERIFICATION OF AUTOMATA MODEL

During simulation processes several situations were verified, by varying the intensity of the traffic (light, regular and crowded). The buffer length is directly influenced by the traffic intensity and will increase or decrease the time needed to pass a segment. System and states diagnose are checked using the formal verification tool. The designed system is proved as deadlock free and provide guaranteed in advance time constraints verification. The behavior properties used to validate the model were composed in such way to verify and guarantee the time interval needed to pass through a segment for each traffic type. After that, the time intervals are used to determine the best routes to follow. The behavior properties intended for formal verification are presented on Table 1, where id_s represents the street segment id and id_c stands for vehicle id. The properties where these acronyms are presented are verified for each segment and vehicle in part.

For simulations and results interpretation some predefined values for traffic parameters were chosen: Three intervals to characterize the traffic load (a segment is considered crowded if the number of vehicles is close to buffer length, bigger than buffer length /2; the traffic is considered fluent and less than buffer length/2; the traffic is fluent, optimal); The minimal speed (v_{min}) is 1.38 m/s, the average speed is 8.3 m/s and maximal speed (v_{max}) is 13.8 m/s; The average vehicles length is 5.2 m.

It must be highlighted that on TABLE 1 the behavior properties formalization considers the input language for UPPAAL which use a simplified version of Timed Computation Tree Logic.

TABLE 1. Behavior Properties to be proved

Prop	Informal description of behavior properties	Formal description of behavior properties
P1	Verify if car(id_c) will pass the street segment in a time interval smaller than a predefine time value T_{max} for "id_s" segment.	E[] car(id_c).Leaving.t - car(id_c).Approaching.t <T_{max}[id_s]
P2	Verify if the time needed to cross a street segment is kept in an interval; this value is used later to analyze and for interpretation;	E[] T_{min}[id_s] < car(id_c).Leaving[id_s].t - car(id_c).Approaching[id_s].t < T_{max}[id_s]
P3	Verify if for street segment "id_s" the load can be bigger than the half of the Buffer.	E[] !street(id_s).len[id_s]) > lengthBuf[id_s]/2
P4	Verify if the segment "id_s" reach the maximal load;	E[] len[id_s] == lengthBuf
P5	Verify if for street segment "id_s" the load can be smaller than the half of the Buffer.	E[] street(id_s).len[id_s]) < lengthBuf[id_s]/2
P6	Verify if deadlock can occur on the model.	A[] not deadlock

Consider a road segment having a buffer with 10 positions at each 30 seconds (the intervals of 2 and 10 seconds were tested also) a vehicle will attempt to enter on the segment and will try to leave it as soon as possible. The **TABLE 2** properties are verified for each simulation scenario. In **TABLE 2**, a selection of behavioral properties is presented.

Before running any simulation some configurations must be done and depending the scale of the analyzed map several car and street objects are instantiated and setup with personal constraints and local variables. For each vehicle a car object is instantiated and the driver behavior can be seen in the time intervals to accelerate, breaking

and to pass the street segment. Each street segment has a corresponding street object. The length and the capacity of the street segment are particular to each object corresponding to the real analyzed map. To cope different situations, like: U turn, parking, leaving to an unmonitored segment, several places was introduced to update the state of the segment.

TABLE 2. Results obtained on formal verification tasks

Property (P)	Safety (S) / Liveness (L)	Verification Result (VR)	Time interval (Ti)	P	S/L	VR	T i	P	S/L	VR	T i
P1	L	Yes	interval of 2 time units	P1	L	Yes	interval of 10 time units	P1	L	Yes	interval of 60 time units
P2	L	Yes		P2	L	Yes		P2	L	Yes	
P3	L	No		P3	L	Yes		P3	L	Yes	
P4	L	Yes		P4	L	No		P4	L	No	
P5	L	No		P5	L	Yes		P5	L	Yes	
P6	S	Yes		P6	S	Yes		P6	S	Yes	

Properties P1 determine for each vehicle the time interval needed to pass the street segment. Property P2 verify if the time needed to pass a segment is staying in a predefined interval specific to each segment. Properties P3, P4 and P5 verify if the congestion segment occurs. If property P4 is true then the bottle neck occur on the segment. Depending on the configuration of the simulation some properties cannot be true. The properties P4 and P5 cannot be true at the same time, is like a double check of the traffic conditions.

The simulations were run on a PC having: 64-bit Windows 7 OS, CPU: Intel Core Duo, 2.20 GHz, 4.00 GB. The result, obtained from UPPAAL, regarding the traffic flow, vehicles passed and verifications results are used later to control the traffic in various situations. From command line the *.xml* file (which contains the models of the vehicles, segments and instantiations) are "called" in order to be verified and validated. The behavioral properties are checked after the model is validated. For a large simulated map an interrogation tool can be proposed in order to send interrogations to UPPAAL and to interpret the results file and to put the information in a way that can be used during the control process.

CONCLUSIONS AND FUTURE WORK

The obtained results confirmed the success of the used approach for modeling and analyzing the urban road traffic. Timed automata, as formalism, and UPPAAL, as software tool, allowed the correct and adapted modeling of this kind of systems and are well adapted for performing formal verification of the complex networked model obtained. Because it allows modular modeling, those formalism and tool make easily the work of the designer. As future work, the team related with this project intends to analyze more complex situations and systematize this approach for implementation in realistic scenarios like, for instance, in a part of a city or in an entire city.

ACKNOWLEDGMENTS

This paper was supported by the project "Development and support of multidisciplinary postdoctoral programs in major technical areas of national strategy of Research-Development-Innovation" 4D-POSTDOC, contract no. POSDRU/89/1.5/ S/52603, project co-funded by the European Social Fund through Sectorial Operational Program Human Resources Development 2007-2013.

REFERENCES

1. E. Seabra, and J. Machado, Using Advanced Simulation Techniques to Improve Industrial Controller's Dependability, INDIN, Lisbon, 2011
2. A. Philippot, Survey on diagnosis of a pick and place benchmark: Special session on diagnosis of Discrete Event Systems: Application on a benchmark, DCDS, 2011.
3. J.C. Campos and J. Machado, Supporting requirements formulation in software formal verification, LADC, Brazil, 2011.
4. J. Machado, E. Seabra, J.C. Campos, F. Soares and C. P. Leão, Safe controllers design for industrial automation systems. CIE'2011, doi:10.1016/j.cie.2010.12.020
5. J. Fitzgerald, P.G. Larsen, K. Pierce, M. Verhoef and S. Wolff, Collaborative Modeling and Co-simulation in the Development of Dependable Embedded Systems, IFM 2010, vol. 6396 pp. 12–26..
6. W. Fokkink, A. Kakebeen and J. Pang, Adapting the UPPAAL model of a distributed lift system, FSEN'07 pp. 81-97, 2007.
7. G. Kunz, E. Perondi and J. Machado, Modeling and simulating the controller behavior of an Automated People Mover using IEC 61850 communication requirements, IEEE -INDIN, Lisbon, Portugal, 2011.
8. G. Frey and L. Litz, Formal methods in PLC programming. SMC 2000, Nashville, October 8–11, 2000.
9. Moon, Modeling programmable logic controllers for logic verification. IEEE Control Systems, 14(2), pp. 53–59, 1994.
10. R. Alur and D.L. Dill, Automata for modeling real-time systems. ICALP'90, England, 1990.
11. P. Ravn, J. Srba and S. Vighio, Modeling and Verification of Web Services Business Activity Protocol, CTACAS, 2010.