Formal Modelling and Verification of the Traffic Light Control System Design with Time-Automata

A. Kamput, and C. Dechsupa

***Abstract*—Modelling and verification are important procedures in traffic light design because they check the model's correctness and determine whether it has desirable properties or not. The traffic light design models must be verified before implementation. Especially in synchronized traffic light intersection systems, where the traffic lights at the multiple intersections require many communication channels for synchronization of the message among intersections and traffic light groups, the traffic light model must be proven meticulously. This paper proposed the modelling and verification of the traffic-right design models by using a time automata named Uppaal. Our templates and frameworks help the modellers create the traffic-right designs in a formal model and can verify the safety properties and structure of the model correctly.**

***Index Terms*—Model checking, Time automata, Software engineering, Traffic right design, Formal modelling**

I. Introduction

THE traffic light control system (TLCS) is important for managing the transportation of urban and metro areas. The intersection control system is crucial to the safety and effectiveness of road users as well as the management of traffic jams. The complexity of the light control system also depends on the number of lenses and the types of intersections: three-intersections, four-intersections, and roundabouts, including the complicated intersection that crosses a railroad.

A real-time traffic light control system has been used in many metropolises, where the controller works based on the measurement of the traffic density on the road to reduce traffic congestion. However, the heterogeneous factors, parameters, and resources of each city are challenging for the designers. Thus, the models of a traffic light control system must be verified meticulously in all possible states to guarantee that the models meet the safety properties and desirable properties.

To verify the TLCS models, the models should be designed by an automated tool or in a specific language that can be simulated or verified by a model checker tool. The creation of TLCS is quite cumbersome, and the designer's competence is required. They should understand the syntax and lexical items of the formal language that is used for modelling and must have competence in a temporal logic expression if they use a model checking technique [1]. As described in the TLCS model, the core process, time constraints, variables, and parameters are components of the formal model TLCS.

Manuscript received Mar 26, 2023; revised Mar 28, 2023.

A. Kamput is an undergrad student at department of computer science, College of computing, Khon Kaen University, Khon kaen,40002, Thailand (e-mail: apipath.k@kkumail.com).

C. Dechsupa, Ph.D. is an assistant professor at department of computer science, College of computing, Khon Kaen University, Khon kaen,40002, Thailand (to provide phone: (+66) 043-009700, 50525; e-mail: chanode@kku.ac.th).

A generality of the formal TLCS model that supports model customization is a valuable attribute for the modellers because it can reduce the time-consuming process to create the TLCS model. As a result of the mentioned obstacles and the designer's needs, this paper proposes a formal model of the TLCS that supports parametrization and customization. The provided models are designed in Uppaal and can be verified in an Uppaal environment. The safety properties and effectiveness of the TLCS can be explored, and the modellers can optimize their own TLCS to reduce vehicle delays and stops efficiently.

The structure of the paper is as follows: Section II describes the background of the TLCS and Uppaal. Section III discusses the related works, and Section IV details the research methodology and experiments. Sections V and VI are the validation and conclusion, respectively.

II. BACKGROUND

A. Traffic light control system

Traffic signal control is a requisite for crossroads in most cities around the world. There are three main signal control categories: fixed time control, actual time control, and advanced time control. The three standard traffic colors of red, green, and yellow represent the Stop, Warning, and Go states, respectively. The form of the signal controller is meticulously determined at the electro-mechanical controller. Phases are representations of the traffic signal aspects that show one or more movements can occur at a moment. A stage is the non-conflicting phase without crossing sections.

Isolated pre-timed and coordinated pre-timed are control mechanisms in which the cycle length, phase plan, and phase times are predetermined and fixed with static values. To optimize traffic congestion management, coordinated signal systems are required to synchronize data among the intersection networks. It can also be envisioned as semi-actual time control or actual time control where detection is provided for the movements.

To ensure safe operation of the intersection, signal timing and controller parameters are determined for managing the right-of-way at a signalized intersection or controlling some of the left-of-way. These parameters may be calculated using the method preferred in the work of [2]. The quality of intersection operation is peculiarly contingent on the relationship between the intersection network and the signal controller settings.

B. UPPAAL

Uppaal is a verification environment that is provided by Uppaal University. It has been used for verifying real-time systems modelled as networks of timed automata. A double-line cycle is an initial state, and a single-cycle represents

ISBN: 978-988-14049-4-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

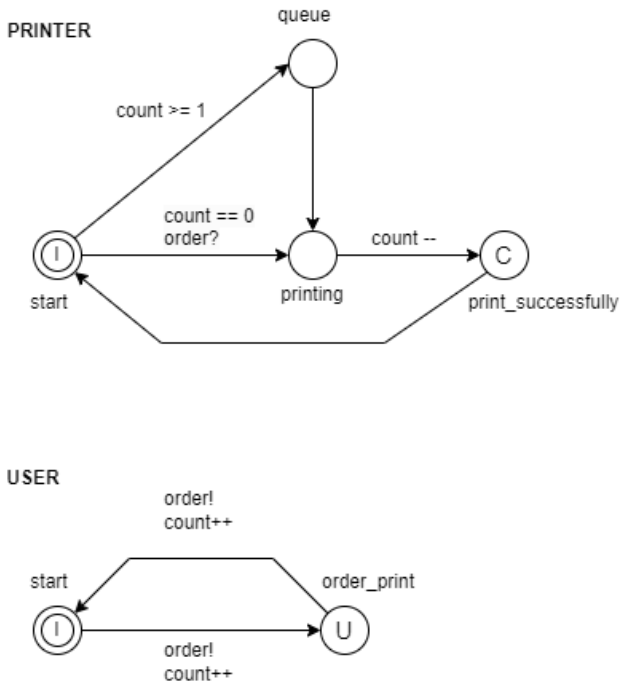

Fig. 1. Uppaal constructs of the printer system.

a system state. Time constraints can be determined for each location in terms of factors that limit what the system can do. There are two special locations: urgent (U) and committed (C), for displaying an event that performs an action or transits a state immediately. A state change is represented by the directed arc that connects two locations. The state change may also depend on a guard condition evaluation, in which the variables, clocks, channels, and constants used in the guard condition expression come from declarations of the model templates. The Uppaal framework allows the modellers to validate their own models in both animation mode and verification mode. An example of the system abstracted by using the time-automata of Uppaal is shown in Fig. 1. The model represents the printer behavior that a data receiving and operation processing is performed only one thread at a moment.

As the Uppaal model shown in Fig. 1, the time-automata that represent a printer system are composed of two templates or processes: *PRINTER* and *USER*. The initial states of them have the label "I" in a double-line cycle. The template *PRINTER* receives printing commands via the global variable *counter*, and its states change from *start* to *queue* when the guard condition evaluation of "counter $\geq$ 1" is true. The state named *queue* in the template *PRINTER* represents the status of the printing queue as existing, and the state named *printing* means a printing status. After finishing each printing job, the printer state will change to be *print_completed*. It is determined to be a *Committed state* with the label "C" because the printer commits the current printing and gets into the initial state immediately (the delay time is zero) to print the next job in the queue.

III. Related work

There are many works [4] that provide methods for designing and verifying the traffic signal control problem. We are focused on the model-oriented approach using model checking approaches [3] because the graphical representation is quite legible and easier to use for both technicians and non-technicians. Applying the model-checking approaches can help the designers verify and simulate the traffic light system model systematically. Vivek et al. [5] proposed the verification fashion for the adaptive traffic signal system by transforming the traffic light scenarios into a finite state machine. NuSMV environments are a verification tool used to check the source of errors and undesirable properties in animation mode and verification mode. This work is merely a guide for verification and how to abstract the simple traffic light model without consideration of real-world constraints.

Alternately, property-oriented approaches may be a viable option for modelling and verifying the complicated traffic signal control problem. Yu et al. [6] proposed a bounded model checking approach to endorse the correctness of a traffic light model. The model checker named BMC4PPTL is used with propositional projection temporal logic (PPTL) for specification and verification of traffic signal control behaviors. The back-end operation of this work is part of the input language of NuSMV. Although the authors provided and demonstrated a simple analysis, it is a practical traffic light control system that can be applied to real-world situations. However, these techniques require the designers to have expertise in PPTL.

Hongli et al. [8] used SysML and NuSMV for performing a system safety analysis. The authors created the traffic light control system as a semi-formal model described in SysML. Next, the model is mapped into a target symbolic model of NuSMV based on their transformation rules. The transformation rules of the controller designed in SysML form can be employed for automating the formal model.

Gleifer et al. [9] provided a double-level model-checking approach, which the model-checking agent programming language used to portray the behaviors of the road junctions. The system behaviors are described as timed automata, for which an extended extra layer of the modelling is performed by using Uppaal, and the implemented system properties are verified in GWENDOLEN.

In advanced time control, the work of [7] provides the techniques for the synthesis and validation of the traffic light controller design. The authors applied Uppaal Stratego, which is a combination of machine learning and model checking techniques, for the synthesis of the optimal controller. The input radar sensors are data for the learning of an optimal controller. These techniques can be applied for analyzing the performance of the loop and static controllers, but the authors demonstrated them with a few scenarios.

IV. Implementation

The contributions of this work are twofold. First, the generic Uppaal constructs are provided in terms of a time-automata network that supports the model parametrization. Last, we propose the properties checking technique for the target Uppaal constructs. The proposed Uppaal constructs consist of the variable and channel declaration, the traffic light controller and the physical traffic light. The details of each construct are as follows:

ISBN: 978-988-14049-4-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

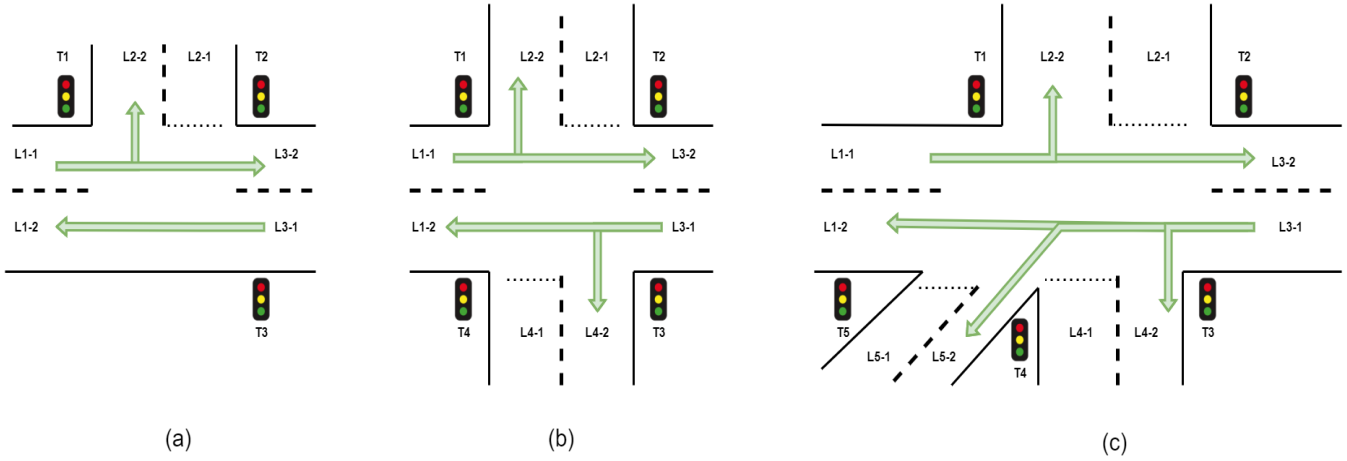

Fig. 2. Basic intersection types and excerpts from signal phases: (a) three-way intersection; (b) four-way intersection; and (c) five-way intersection.

```
int yl_count = 0;       //time counter for green
int yl = 5;             //maxim time uit for green
int l1_r = 0;           //maxim time uit for red
int l1_g = 5;           //maxim time uit for green
int l1_count_r =  0;    //time counter for  red
int l1_count_g =  0;    //time counter for green
//variables to sync data between processes
broadcast chan step_one,step_two,step_three;
broadcast chan lane_s,lane_b,next_step;
```

Fig. 3. Excerpt variables and channels declaration.

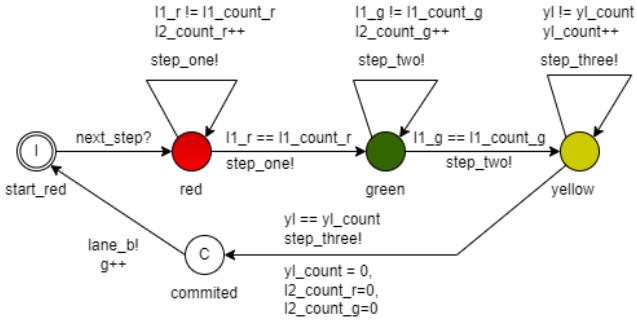

Fig. 4. Uppaal construct of the physical RGY traffic light.

A. Declaring the model's variables and channels

Variables and channels used in the Uppaal model have to be declared in the declaration sector. Most of the variables are used as counters and collect the time constant of each traffic light sign. While the channels are used for data synchronization between the controller and the physical traffic light. Supporting the mentioned parametrization purpose, the designers can adjust the values of each variable in order to determine the cycle length, green interval, and red interval arbitrarily. Excerpt variables and channels declaration are shown in Fig. 3.

B. Creating the Uppaal template of physical traffic light

Components of the Uppaal construct of the traffic light system with Red Green Yellow are portrayed in Fig. 4. Each light pole occupies the three states represented by Uppaal locations as follows:

- *Start_red* is an initial state.
- *Red* is a red-light state.
- *Green* is a green-light state.
- *Yellow* is a yellow-light state.

Declared in the declarations section of the Uppaal model, variables and channels used in this template appear on the edge. The channel *next_step* is a channel used for getting the signal from the controller(s), and the channel *lane_b* is a channel that is used to respond with the signal to the controller when the traffic light state is complete. For each RGY state change, the time-counter and the global clocks are used for guard condition expression. For instance, *r1_count_r* is the red-light clock counter, which is defaulted to zero, and *l1_r* is the global variable storing the static maximum clock of the red light. These variables are expressed as the guard condition *l1_count_r != l1_r* and as assigning a value increasing by one with *l1_count_r++* at the location *red*. It can be seen that each traffic light state change has the guard condition and time counter.

C. Creating the traffic light controller

The Uppaal controller is designed for supporting the three ordinary types of intersections: three-way intersections, four-way intersections, and five-way intersections. The graphical representation of the intersections is shown in Fig. 2. We have designed the Uppaal parametrizable controller for all interaction types to simulate and validate the signal control scenarios. Due to the bulkiness of the model, we separate the Uppaal controller into three parts, following the intersection types.

1) Controller of the three-way intersection: The controller of the three-way intersection is also partitioned into the stage-control and lane-control (phases controller) components. The state-control is shown in Fig. 5 and the phase-control is shown in Fig.6. Let's consider the stage in Fig. 2(a), the traffic light controller is working in the phase of three-way flow, in which two lanes can go straight and one turns left on red. Thus, the variable *fm3_count* is two, and the controller mandates the physical traffic light *T1* and *T3* to be on green. The physical traffic light *T1* controls the lane *L1-1* and *L2-2*. In other words, the physical traffic light *T2* that controls the lane *L2-1* is blocked on red alone.

2) Controller of the four-way intersection: The controllers of the four-way intersection are shown in Fig. 7 and 8,

ISBN: 978-988-14049-4-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

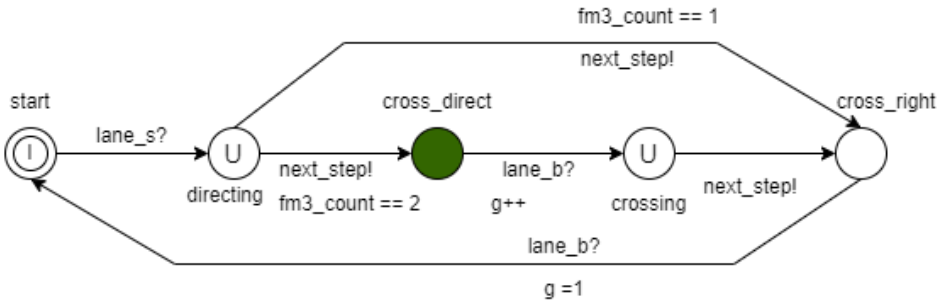

Fig. 5. UPPAAL construct of the stage controller of three-way intersection.

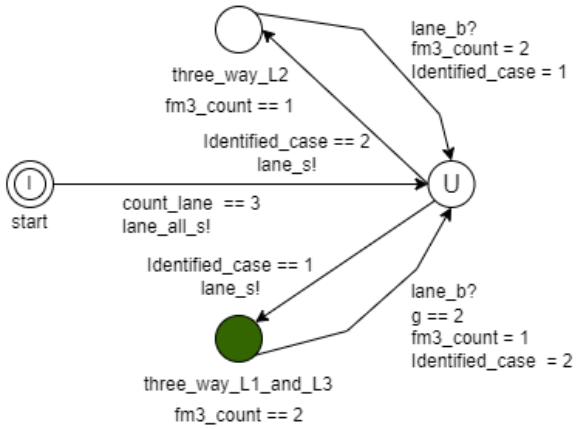

Fig. 6. UPPAAL construct of the phase controller of three-way intersection.

which represent the stage control and the phase control, respectively. As the stage shown in Fig. 2(b), the stage of the four-way intersection shows the three-way flow in which the lanes *L1-1, L1-2, L3-1, L3-2, L2-2* and *L4-2* are active, whereas the lanes *L2-1* and *L4-1* are blocked. It means that there are two phases activated with six active lanes. This state is represented by the green location in Fig. 7,which the variable *fm4_count* is two.

3) Controller of the five-way intersection: The controllers of the stage control and the phase control of a four-way intersection are shown in Fig. 9 and 10 respectively. As the stage in Fig. 2(c), the physical traffic light *T1* and *T3* is on green while the others are on red. This state is represented by the green location in Fig. 9, where the variable *fm5_count* is two and the identified case is one shows that the traffic light *T1* and *T3* are on green and the others are on red.

D. Validation

The Uppaal model is validated in both simulation mode and verification mode. All the phases and stages of each type of intersection are checked scrupulously. The desirable properties are expressed in CTL and verified in the verification mode. Due to space limitations, we describe the excerpt CTLs applied to the proposed model in Table I.

V. Conclusion

The model-checking technique is an alternative way of animating and verifying a concurrent system to prove that it is deadlock free and meets the desired properties or not. This work provides the formal traffic light templates and verification techniques by using a time-automata network of Uppaal. The contributions are twofold: 1) creating the generic traffic light model supporting parametrization and 2) providing the property checking based on the proposed model. The Uppaal constructs comprise three main parts: declaration, controller, and physical traffic light. The provided Uppaal models advocate the three-way intersection, the four-way intersection, and the five-way intersection. We validated the model in both simulation mode and verification mode by showing that the desirable properties are expressed in CTL. All the phases and stages of each type of intersection are verified scrupulously. We observed that the proposed model can represent the intersection behaviors realistically, and the model can also be customized and applied to other phases of a physical traffic light. The Uppaal environment can verify undesirable and desirable properties of the model by using our properties written in CTL. However, the proposed Uppaal model does not consider the pedestrian crossing or railroad crossing. These constraints will be added to the model in the future, and the advanced time control applied by AI is our ongoing work.

References

[1] C. Baier and J.-P. Katoen, Principles of Model Checking. MIT Press, 2008.
[2] Y. Ryabokon, “The method of determining the number of phases in the traffic light cycle on the allowable intensity of conflicting flows,” *Transportation Research Procedia*, 2017, pp. 571-577.
[3] B. Christel, and J. Katoen, Principles of model checking, MIT press, 2008.
[4] E. Myungeun, and B. Kim, ”The traffic signal control problem for intersections: a review,” *European transport research review*, 2020, pp. 1-20.
[5] V. Vivek, S. Gugwad, and S. Singh, ”Modeling and Verification of Agent based Adaptive Traffic Signal using Symbolic Model Verifier,” arXiv preprint,2012, pp.1208.3461.
[6] Y. Bin, Z. Duan, and C. Tian, ”Bounded model checking of traffic light control system,” In *Electronic Notes in Theoretical Computer Science*, 2014, pp. 63-74.
[7] E, Andreas Berre, C, Huang, J. Kildebogaard, H. Lahrmann, K. G. Larsen, M. Muniz, and J. H. Taankvist, ”Uppaal stratego for intelligent traffic lights,” In *12th ITS European Congress*, 2017, pp. 1-10
[8] W. Hongli, D. Zhong, T. Zhao, and F. Ren. ”Integrating model checking with SysML in complex system safety analysis,” *IEEE Access*, 7, 2019, pp. 16561-16571.
[9] A. G. Vaz, L. Dennis, and M. Fisher. ”A double-level model checking approach for an agent-based autonomous vehicle and road junction regulations,” *Journal of Sensor and Actuator Networks*, 10, no. 3 (2021): 41.
[10] G. Payal, D. V. Gadre, and T. K. Rawat, ”Real Time Traffic Light Control System (Hardware and Software Implementation),” *International Journal of Electronic and Electrical Engineering*, 2014, pp. 505-510.

ISBN: 978-988-14049-4-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

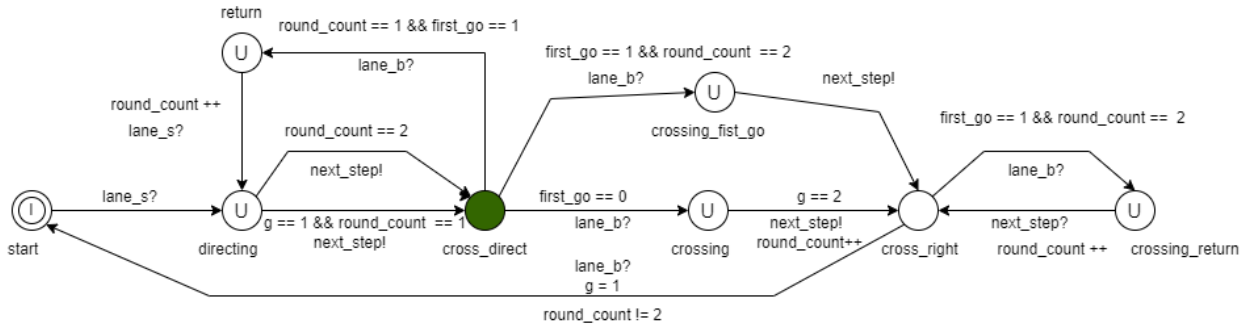

Fig. 7. UPPAAL construct of the stage controller of four-way intersection.

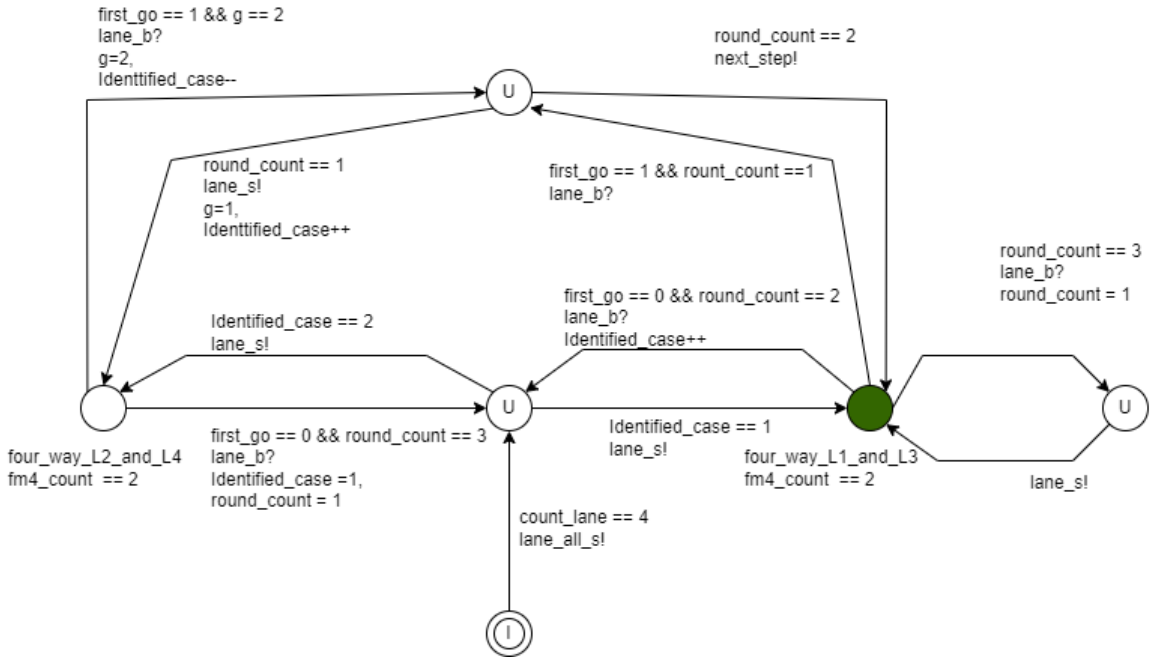

Fig. 8. UPPAAL construct of the phase controller of four-way intersection.

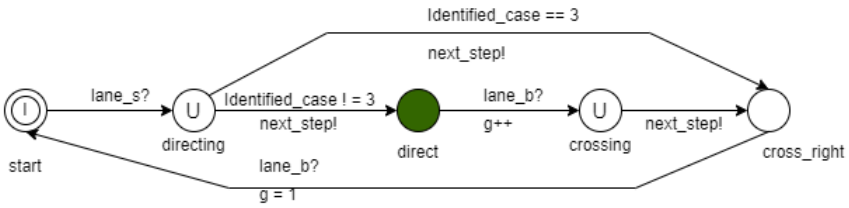

Fig. 9. UPPAAL construct of the stage controller of five-way intersection.

ISBN: 978-988-14049-4-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

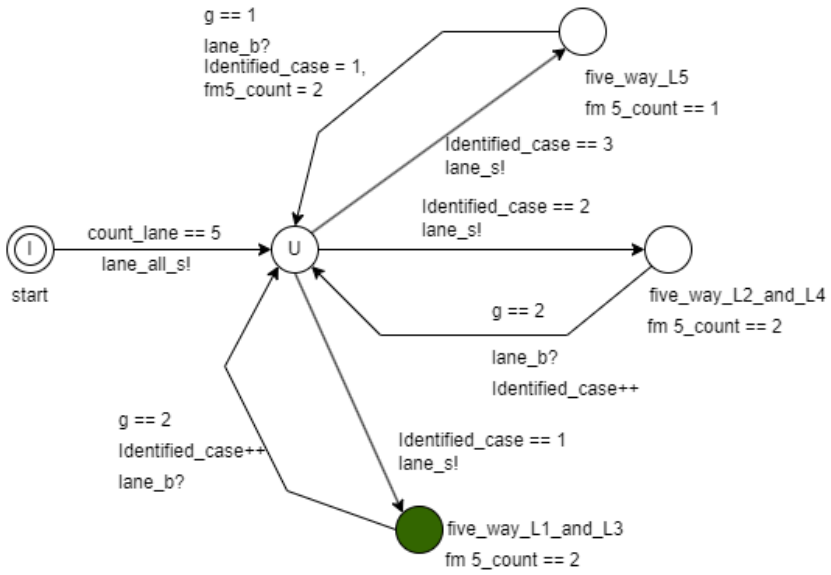

Fig. 10. UPPAAL construct of the phase controller of five-way intersection.

TABLE I
EXCERPT PROPERTIES AND CTLS FORMULA APPLYING TO THE PROPOSED UPPAAL MODEL

No.	Properties	CTLS
1	The system is guaranteed to be free from deadlock.	E<> not deadlock
2	For three-way intersection checking , none of the observed events resulted in L1 and L2 signals turning green concomitantly.	E<> not(PC3_L1_direct.green and PC3_L2_direct.green
3	For three-way intersection checking , none of the observed events resulted in L1, L2 and L3 signals turning green concomitantly in the direction of straight movement.	E<> not(PC3_L1_direct.green and PC3_L2_direct.green and PC3_L3_direct.green)
4	For three-way intersection checking, there were no observed events resulting in L1, L2, and L3 signals turning green concomitantly in the direction of right-turn movement.	E<> not(PC3_L2_direct.green and PC3_L3_right.green)
5	For four-way intersection checking , none of the observed events resulted in L1, L2, L3 and L4 signals turning green concomitantly in the direction of straight movement.	E<> not(PC4_L1_direct.green and PC4_L3_direct.green and PC4_L2_direct.green and PC4_L4_direct.green)
6	For four-way intersection checking, there were no observed events resulting in L1, L2, L3, and L4 signals turning green concomitantly in the direction of right-turn movement.	E<>not(PC4_L1_right.green and PC4_L3_right.green and PC4_L2_right.green and PC4_L4_right.green)
7	For four-way intersection checking, what events would cause signals L1 and L3 to turn green simultaneously in the direction of right-turn movement, and signals L2 and L4 to turn red simultaneously.	E<> PC4_L1_right.green and PC4_L3_right.green and PC4_L2_right.start_red and PC4_L4_right.start_red
8	For three-way intersection checking, there were no observed events resulting in L1 and L3 signals turning green concomitantly in the direction of straight movement, and L2 and L4 signals turning green concomitantly in the direction of right-turn movement.	E<>not(PC4_L1_direct.green and PC4_L3_direct.green and PC4_L2_right.green and PC4_L4_right.green)
9	For five-way intersection checking, there were no observed events resulting in L1, L2, L3, L4, and L5 signals turning green concomitantly in the direction of straight movement.	E<>not(PC5_L1_direct.green and PC5_L3_direct.green and PC5_L2_direct.green and PC5_L4_direct.green and PC5_L5_direct.green)
10	For five-way intersection checking, there is no event that can be observed to cause signals L1, L2, L3, L4, and L5 to turn green simultaneously in the direction of right-turn movement.	E<>not(PC5_L1_right.green and PC5_L3_right.green and PC5_L2_right.green and PC5_L4_right.green and PC5_L5_right.green)

ISBN: 978-988-14049-4-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)