Intelligent Traffic Control Model Checking Using UPPAAL

Mohit Awachar
Department of Information Technology
National Institute of Technology Karnataka
Surathkal-575025, India
mohit.191it231@nitk.edu.in

Anshul Patel
Department of Information Technology
National Institute of Technology Karnataka
Surathkal-575025, India.
anshulpatel.191it208@nitk.edu.in

Rakshit Kulkarni
Department of Information Technology
National Institute of Technology Karnataka
Surathkal-575025, India
rakshitkulkarni.191it245@nitk.edu.in

Kiran Kumar J M
Department of Information Technology
National Institute of Technology Karnataka
Surathkal-575025, India
kirankumarjm.191it126@nitk.edu.in

***Abstract*—This paper focuses on model checking of a Traffic Light System. This paper is an extension of the paper Uppaal Stratego for Intelligent Traffic Lights published in France in 2017. In the base paper the plan was to find an optimal strategy for minimum traffic. Here the main focus is on model checking of traffic light systems and verifying if all the states are reachable and if all the safety requirements are satisfied. Uppaal is the tool used for model checking in this work. It is an integrated tool for modeling, simulation and verification of real-time systems(in this case the traffic light system).**

Keywords— traffic lights, model checking, uppaal , verification and validation.

I. INTRODUCTION

Traffic lights also called traffic lamps or signal lights have the responsibility to manage the flow of traffic at junctions or intersections of roads. A typical traffic light has three colors namely red indicating to stop, green indicating the vehicle can cross the junction and yellow or orange means the signal is about to switch to green or red. Traffic lights play a very crucial role in flow of traffic every day and it prevents or at least reduces many vehicle crashes that can happen otherwise. It ensures a safe and orderly flow of traffic on roads and highways. In addition, traffic lights also ensure safety of pedestrians crossing the road. The signal ensures that pedestrians also get a share or road without worrying about safety. This paper focuses on the model checking and verification of safety conditions of traffic light systems.

Model checking is a method for checking if a finite-state model of a system satisfies the specifications provided. Traffic-Light Control System is the system under consideration with respect to this paper. Traffic-Light system being a safety critical system, it's failure can lead to heavy loss of property as well as life. To solve this problem algorithmically, both models of the Traffic Light System and it's specifications are formulated in precise mathematical language. Now that the problem is formulated as a task in logic, the model checking tool needs to verify if the logical property is satisfied for the given logical formula. A simple model-checker can be a tool verifying whether a formula in propositional logic is satisfied by a system or not.

Symbolic Model Checking(SMC) is a formal verification technique which does not require any user assistance. In SMC, it is possible to verify an implementation by modelling it as an finite-state transition graph to check if it satisfies given specifications. This is done by giving some properties of temporal logic.

Many tools are available for model checking like NuSMV, PAT, Prism, etc. In this implementation we have used Uppaal stratefor for model checking. It is an integrated tool for modeling, verification and validation of real-time systems. The system is modelled as a network of timed-automata. More information about this tool is discussed in the methodology part.

II. LITERATURE SURVEY

In this paper [1] the authors have done a simulation of traffic in sumo and use uppaal as backend tool to find the optimal strategy to minimize the waiting time.

In this paper [2] the authors have extended the work in [1]Uppaal Stratego for Intelligent Traffic Lights. The authors considered multiple traffic lights and used UPPAAL Stratego to find the optimal stratego by bringing a coordination between data collected by the signals to reduce the traffic.

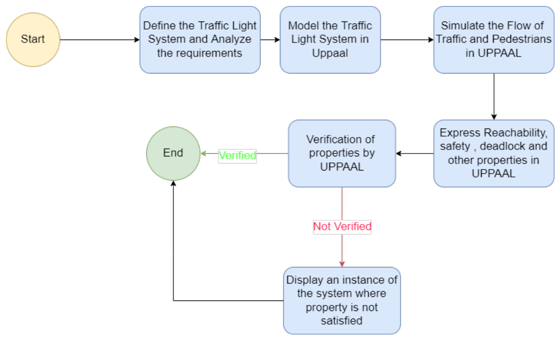
All the papers were focusing on finding the optimal strategy to find the optimal strategy to reduce traffic. None of the papers did the model checking of the traffic light system and verified it's safety and reachability properties. We aim to do the same in this paper.

III. Problem Statement

Intelligent Traffic Control Model Checking using UPPAAL.

IV. Proposed Work

1) Define the Traffic Light System
2) Model the Traffic Light System in UPPAAL
3) Simulate the Flow of traffic and Pedestrians in UPPAAL
4) Express Reachablitiy, Safety, Deadlock and other properties in UPPAAL
5) Verification of Properties in UPPAAL



Start
Define the Traffic Light System and Analyze the requirements
Model the Traffic Light System in Uppaal
Simulate the Flow of Traffic and Pedestrians in UPPAAL
Express Reachability, safety , deadlock and other properties in UPPAAL
Verification of properties by UPPAAL
Verified
End
Not Verified
Display an instance of the system where property is not satisfied

Fig. 1: The Traffic Light.

V. Methodology

Let's look at the modelling tool Uppaal Stratego first.

Uppaal Stratego : Uppaal is an integrated tool for modeling, simulation and verification of real-time systems. This tool is developed by joint contribution of the Basic Research in Computer Science at Aalborg University in Denmark and the Department of Information Technology at Uppsala University in Sweden. Typical applications are systems where timing is important such as real-time controllers and communication protocols.

Uppaal consists of three major parts

- **Description Language :** This is a non-deterministic language with different data types. It serves as a design language to describe our system behavior as a network of automata. The language also supports clock and data variables.
- **Simulator :** This is a validation tool which enables examination of possible dynamic execution of the system under consideration.
- **Model-checker :** This checks for the reachability properties by exploring the state of the system. It also checks the invariant properties.

In the Uppaal application one can see three portions

- **Editor :** In the editor section of UPPAAL the user can define various states of his system. The user can create a template for a particular subsystem and define various states and transition among them for the subsystem. In the global declarations one can define global variables. In the system declarations, users can define multiple or single instances of the templates created.The various systems can be synchronized with the help of channels. The time can be monitored with the help of clocks.
- **Simulator :** In this section, the user can see all the instances of the subsystem created by him. In the top left corner the possible transitions are shown. In the bottom right corner the detailed history of all transitions is shown. In the top right corner (i.e. the major portion) all the state diagrams of all instances are shown and the current states are highlighted. Other than this all global and local variables are also visible.
- **Verifier :** In this section the user can insert the properties he/she is interested to verify for the system.

The Traffic Light system : We are considering a traffic light system for model checking. The system is an intersection of two perpendicular roads and a traffic light present at the intersection. As shown in the figure below, the two perpendicular roads NS and EW intersect at a point TL(Traffic Light). The system also has crossings for the pedestrians.

The traffic light has three phases namely green , red and yellow for both the lanes. Following are the rules we follow for the traffic light.

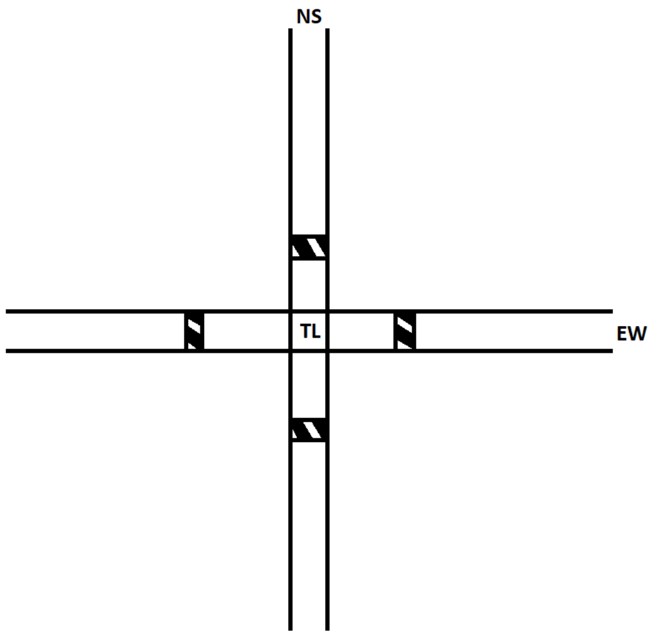


Fig. 2: The Traffic Light.

- When a lane is red the vehicle can't cross. When the lane is green, the vehicle has a choice to cross or wait.
- Once a lane(NS or EW) is made green, the lane remains green as long as some vehicle is waiting in the lane
- Once both the lanes are red the pedestrians can cross the road. The lanes will remain red until all the pedestrians

cross the lane.
- The lane have a fixed timer of 5 seconds for the yellow signal.
- The traffic lights go in order green-yellow-red-yellow-green.

The UPPAAL Model : We define 4 templates for our system in UPPAAL.

1) The Traffic Light

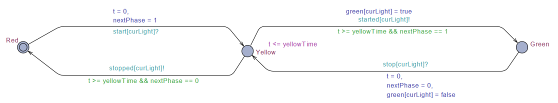


Fig. 3: The Traffic Light.

Parameters : curLight and yellowTime
Local Declarations : clock and nextPhase
Possible States :

- Red : The vehicle can't cross and the pedestrians are free to cross the road.
- Yellow : This is the phase between green and red and is encountered two times in a cycle. The phase has an invariant that the subsystem must remain in this state for 5 seconds each time.
- Green : The vehicles in the current lane can cross and the pedestrians have to wait.

At the start the signal is in Red state. The signal can move to Yellow state from here once it receives the signal (from channel start[curLight]) from the controller. During this transition the controller updates t=0 and nextPhase=1. In the Yellow state, the signal must stay for yellowTime and then the signal has two options namely Green or Red State. When nextPhase is one the signal moves to Green state and moves to Red state otherwise. During these transitions, a signal(started[curLight] or stopped[curLight]) is sent to the controller. When the TrafficSignal is in Green state, it can move to Yellow state once it receives the signal(from channel stop[curLight]) from the controller. During this transition, time and nextPhase are updated to 0.

2) The Vehicle

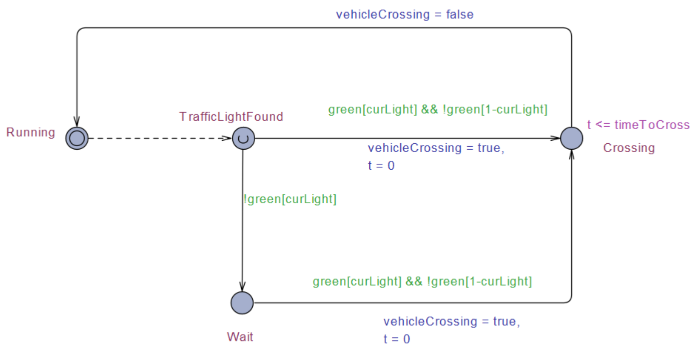


Fig. 4: The Vehicle.

Parameters : curLight and timeToCross
Local Declarations : clock
Possible States :

- Running : The vehicle is freely running on free roads.
- TrafficLightFound : The vehicle encountered a traffic light. Wait : The vehicle is waiting in front of the traffic light.
- Crossing : The vehicle is crossing the junction.

At the start, the vehicle is in Running state. When the vehicle encounters a traffic light it is shifted to TrafficLight Found state. This state is marked as urgent meaning one must take a transition here. If the lane is green the Vehicle moves to Crossing state else it moves to Wait state. The vehicle can stay in the Crossing state for time less than timeToCross and after this it moves to Running state again. The Vehicle can move from Wait to Crossing when the lane is green.

3) The Pedestrian

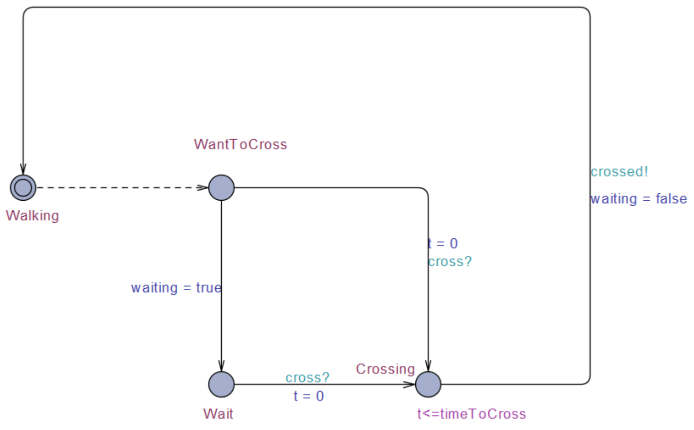


Fig. 5: The Pedestrian.

Parameters : curLight and timeToCross
Local Declarations : clock
Possible States :

- Walking : The pedestrian is walking freely on a footpath.

- WantToCross : The pedestrian encounters a road junction.
- Wait : The pedestrian is waiting for vehicles to pass to cross the junction.
- Crossing : The pedestrian is crossing the junction.

Initially the Pedestrian is in Walking state. When he/she encounters a junction he/she changes to WantToCross state. At this stage if we receive a cross signal from the controller the Pedestrian can move to Crossing State. Else the Pedestrian moves to Wait state. From the Wait state, The Pedestrian moves to Crossing state on encountering cross signal from the Controller. The Pedestrian can stay in Crossing state for less time less than timeToCross and then moves to Walking state again. When this transition is in progress a signal is sent to the controller that the Pedestrian has crossed the junction.

4) The Controller

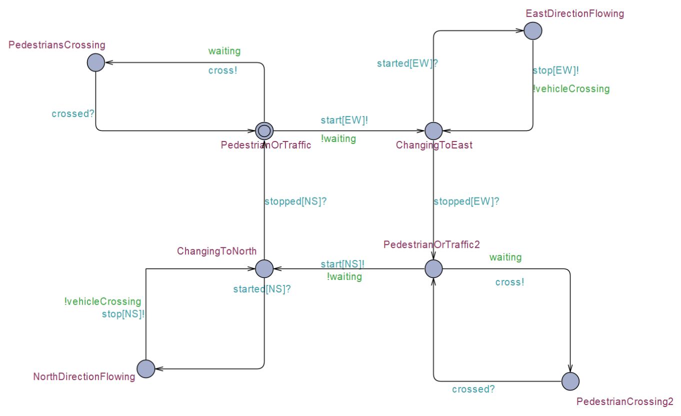


Fig. 6: The Controller.

Parameters : None
Local Declarations : None
Possible States :

- PedestriansCrossing / PedestriansCrossing2 : The pedestrians are crossing the junction.
- PedestrianOrTraffic / PedestrainOrTraffi2 : The controller can allow Pedestrians or vehicles to cross
- ChangingToEast : The flow of vehicles along EW lane can start/stop
- ChangingToNorth : The flow of vehicles along NS lane can start/stop
- EastDirectionFlowing : The EW lane is active
- NorthDirectionFlowing : The NS lane is active

The Controller is initially in PedestrainOrTraffic state. If a Pedestrian is waiting to cross, the Controller moves to PedestrainsCrossing state and when the Pedestrian crosses the road, the Controller returns to PedestianOrTraffic state. The other option in this state is to move to ChangingToEast state. When this transition is made a signal is sent to TrafficSignal. From ChangingToEast state, the Controller can move to EastDirectionFlowing allowing the flow of traffic in the EW lane. When no vehicle is crossing and waiting, the Controller can return to ChangingEast. The other option at this stage is to move to PedestianOrTraffic2 state. This state is similar to PedestrianOrTraffic state. From there the Controller moves to ChangingToNorth state. Again this state is similar to ChangingToEast only difference being here the flow of traffic is in NS direction. From this state, when flow of traffic is complete in NS direction, the Controller moves to its initial state(i.e PedestrianOrTraffic).

Verification of Properties in Verifier
After the system is modelled in editor and simulated in simulator, the verification of reachability and safety properties is done in the verifier. More about these properties is discussed in the RESULT section.

VI. RESULTS AND FUTURE WORK

In the simulator, we have the following subsystem.

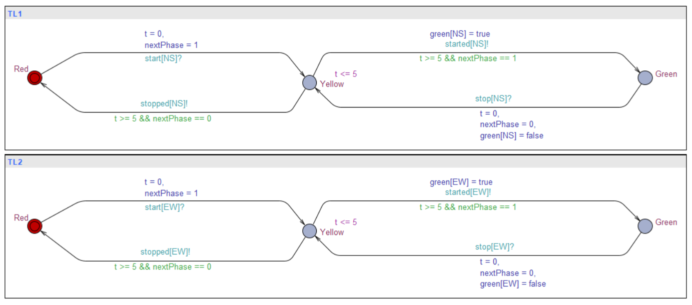


Fig. 7: The two traffic lights namely NS and EW

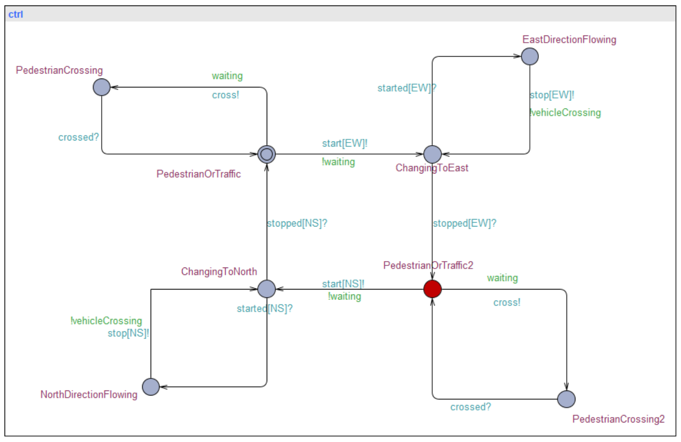


Fig. 8: The controller

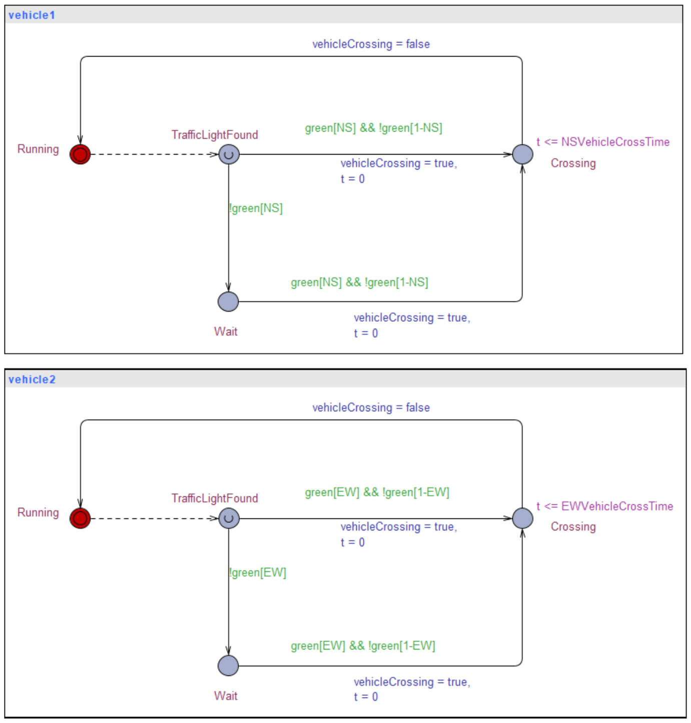


Fig. 9: The Vehicles

Two vehicle types are vehicle1 flowing in the NS direction and vehicle2 in the EW direction.

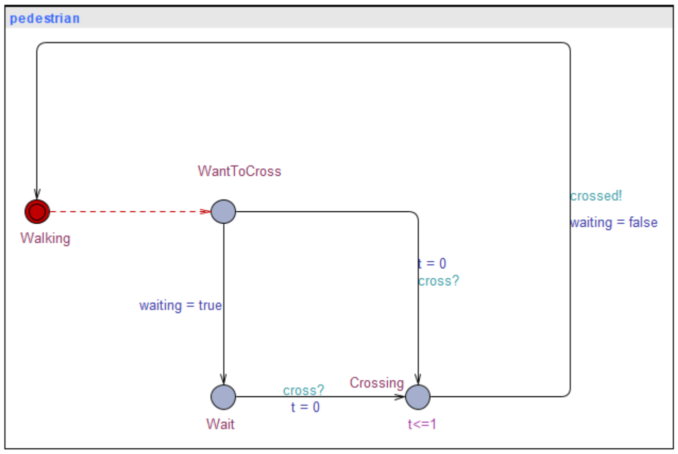


Fig. 10: The Pedestrian

Finally we have the pedestrian.

We verified the following properties for the system.

Reachability Properties :

1) For traffic Lights(TL1 and TL2)
   a) E<> TL1.Red
   b) E<> TL1.Green
   c) E<> TL1.Yellow
   d) E<> TL2.Red
   e) E<> TL2.Green
   f) E<> TL2.Yellow

   In these properties we aim to verify if all the states are reachable. So for each state of both the traffic signals we find if a path exists to reach there. All the above queries are satisfied.

2) Pedestrian
   a) E<> pedestrian.Walking
   b) E<> pedestrian.WantToCross
   c) E<> pedestrian.Wait
   d) E<> pedestrian.Crossing

   For pedestrian all the states (i.e. Walking, WantToCross, Wait and Crossing are reachable). All the above properties are verified.

3) Vehicle
   a) E<> vehicle1.Wait
   b) E<> vehicle1.TrafficLightFound
   c) E<> vehicle1.Crossing
   d) E<> vehicle1.Running
   e) E<> vehicle2.Wait
   f) E<> vehicle2.TrafficLightFound
   g) E<> vehicle2.Crossing
   h) E<> vehicle2.Running

   For both the vehicles, all the states are reachable. All the above properties are verified.

4) Controller
   a) E<> ctrl.ChangingToNorth
   b) E<> ctrl.ChangingToEast
   c) E<> ctrl.PedestrianOrTraffic
   d) E<> ctrl.PedestrianOrTraffic2
   e) E<> ctrl.PedestrianCrossing
   f) E<> ctrl.PedestrianCrossing2
   g) E<> ctrl.NorthDirectionFlowing
   h) E<> ctrl.EastDirectionFlowing

   For the Controller, all the possible states are reachable so all the above properties are verified.

Deadlock Property :

A[] not deadlock

This means we did not get deadlock at any time. The property is also satisfied.

Safety Properties :

1) **A[] not (pedestrian.Crossing and (TL1.Green and TL2.Green))**

   Pedestrian Crossing and both Traffic Lights are green; this condition is not possible at all times. The property is verified.

2) **A[] not (TL1.Green and TL2.Green)**

   At all times both traffic lights can't be green. The property is verified.

3) **A[] not (TL1.Yellow and TL2.Yellow)**
At all times both traffic lights can't be yellow. The property is verified.

4) **A[] not ((vehicle1.Crossing or vehicle2.Crossing) and pedestrian.Crossing)**
A pedestrian can't cross when either of the vehicle is crossing the road. The property is verified.

5) **A[] not (vehicle1.Crossing and vehicle2.Crossing)**
The vehicles in perpendicular lanes can't pass at the same time. The property is verified.

6) **A[] not (vehicleCrossing and (ctrl.ChangingToEast or ctrl.ChangingToNorth))**
When a vehicle is passing through the junction, the controller must not change the traffic signal state. The property is verified.

Other Properties :

1) a) **E<> vehicle2.Wait imply vehicle2.Crossing**
   b) **E<> vehicle1.Wait imply vehicle1.Crossing**
When a vehicle is waiting, it will get to cross eventually. The property is verified.

2) **E<> pedestrian.WantToCross imply pedestrian.Crossing**
When a pedestrian wants to cross, he/she will get a chance to do so eventually. The property is verified.

3) **E<> (ctrl.NorthDirectionFlowing or ctrl.EastDirectionFlowing) imply (ctrl.PedestrianCrossing or ctrl.PedestrianCrossing2)**
When vehicles are crossing then after some time eventually, pedestrians will get a chance. The property is verified.

4) a) **E<> ctrl.EastDirectionFlowing imply ctrl.NorthDirectionFlowing**
   b) **E<> ctrl.NorthDirectionFlowing imply ctrl.EastDirectionFlowing**
When one of the lanes is active, after some time eventually the other will be active. The property is verified.

5) **A[] not (TL1.Red and TL2.Red)**
At any time both traffic lights can't be red. But in this system they can be(when the pedestrians are crossing). So this property is **not satisfied** and the corresponding trace is visible in the simulator.

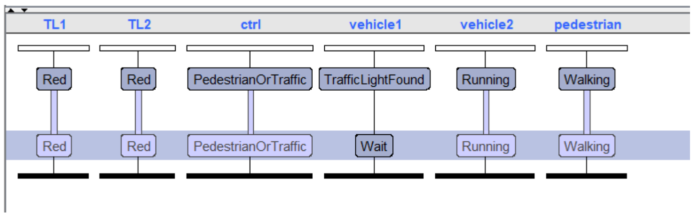


Fig. 11: The trace for both red signals

VII. CONCLUSIONS AND FUTURE WORK

The traffic light system is modelled in UPPAAL with the help of state diagrams. The simulation is done in the simulator and one can manually move from one state to next according to simulation purposes. All the safety requirements, reachability properties and some other properties are properly analyzed and verified against the system using query language. The system passed all the safety and reachability requirements and no deadlock condition was encountered. The project was a very good learning experience for us. We get to learn about UPPAAL, verification and validation in Software Engineering.

In the future, we plan to take a more complex traffic system with multiple roads at the intersection and allow different types of vehicles for modeling the system.

VIII. REFERENCES

[1] Eriksen, Andreas Huang, Chao Kildebogaard, Jan Lahrmann, Harry Larsen, Kim Muniz, Marco Taankvist, Jakob. (2017). Uppaal Stratego for Intelligent Traffic Lights.

[2] B K, Thamilselvam Kalyanasundaram, Subrahmanyam Rao, M. (2019). Coordinated Intelligent Traffic Lights using Uppaal Stratego. 789-794. 10.1109/COMSNETS.2019.8711457.

About Uppaal :

- https://en.wikipedia.org/wiki/Model_checking
- https://uppaal.org/features/

Download UPPAAL at :

- https://uppaal.org/downloads/