# Modeling and Verification of Car Parking System

Hadiqa Alamdar Bukhari
School of Electrical Engineering and Computer Science
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
hbukhari.bese16seecs@seecs.edu.pk

Dr. Sidra Sultana
School of Electrical Engineering and Computer Science
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
sidra.sultana@seecs.edu.pk

***Abstract*— Formal modeling and verification help in achieving safety related concerns in real time systems. Car parking system is modeled and verified in this paper to ensure the non-collision and the parking of a car which is both time and space efficient. A detailed simulation of the model is presented and described.**

***Keywords-Uppaal; system verification; system modeling; car parking.***

## I. INTRODUCTION

Metropolitan cities are dealing with increasing traffic and in turn an increase in parking garages. With the increase in the use of automated systems and the introduction of Internet of Things, more cities are making use of smart car parking systems to deal with the common problem of finding a vacant parking space. Smart car parking systems deal with directing cars to an empty parking spot all the while keeping a record of which parking spots are free.

Car parking systems are safety critical and need to be modeled and verified before they are put into use. A single mistake may lead to damage of a person's car and waste of time due to an inefficient system [1]. These systems are incredibly complex and make use of expensive hardware. Therefore, it is essential that such systems are properly modeled, tested and verified on software before they are implemented on hardware [2]. System verification is of paramount importance [3] and advancements in technology have allowed us to optimize and check the safety of a system on a computer before it is constructed in the real world.

In this paper we modeled the car parking system using Uppaal model checker. The verified model with the timed automata can be used to implement a real time car parking system. We verified the safety, deadlock freeness, reachability, liveness, mutual exclusion, utility and fairness, which Uppaal's inbuilt verification module allowed us to do easily [4].

Following the introduction, in Section II a literature review is given where other similar projects are discussed. In Section III the system overview consisting of a model for car and a model for lane is described and the timed automata of these two models are also shown. The details of the Uppaal model checker and reasons behind why it is used are also discussed in this section. In Section IV system is verified and the verification properties are described. Finally, in Section V the conclusion and further improvements to the system are detailed.

## II. LITERATURE REVIEW

A number of different car parking systems have been made in the past. In this section we will be discussing a few of these systems.

In [5], a car parking system is developed where the presence of a car causes the gates of a parking lot to open and the number of cars in the parking lot are displayed on an LCD. Here the authors focused mainly on the hardware aspects and the modeling and simulation were done on hardware rather than software.

In [6], a similar system to [5] is developed but the car is allotted the closest parking spot by judging the distance of the car from the entrance or exit of the parking lot. The authors in [7], aim to use an RFID and an infrared sensor based parking system. Here hardware modules are used to verify the correctness of the system which can be more expensive than using a modeling and verification software. In our project we significantly cut costs by modeling and verifying our system on the Uppaal model checker instead.

In [8], a web application based car parking system is made where a camera is used to check the availability of a parking spot and if a parking spot is free the user is notified. In [9], the authors have described a mobile application system that uses infrared sensors to find and allocate a parking spot.

In the above mentioned works, modeling and verification of the system was highly dependent on hardware modules. Hardware is not as reliable however, and it can be very hard to check all the verification properties on it. We chose to use Uppaal model checker in our project to ensure that all verification properties like reliability or deadlock freeness etc. are fulfilled.

In [10], the authors used Vienna Development Method-Specification Language (VDM-SL) to develop and verify a graph-based model. This model is used to find nearest empty parking spots. This is the most relevant project however, the simulator which we have used in our project, Uppaal is more versatile than VDM-SL and can be used to understand the traces to further correct the model, or to draw conclusions [11].

## III. SYSTEM MODELING

In this section we give a system overview. Details on the Uppaal model checker are given and the timed automata of car parking system is detailed.

 ISBN: 978-1-61208-752-8 

### A. System Overview

As shown in Fig. 1, we modeled two automata, one for the car and one for the street on which the car looks for a parking space.

The cars can move on a street with two lanes and their sensors query a parallel automaton that models the street layout. The car looks for a free parking space and if there is one present on either lane, it moves into the available parking space and performs parallel reverse parking. After 40 seconds the cars can move out of the parking space and move forward to the end of the street. The street is 5 meters long and it has 2 suitable parking spaces in each lane. Two cars cannot park in the same parking spot at once.

The parallel automata communicate through shared channels. The cars can park multiple times in the street so long as there is a parking space available and the end of street has not been reached. The car can go back to the start of the street after it exits the street as well.

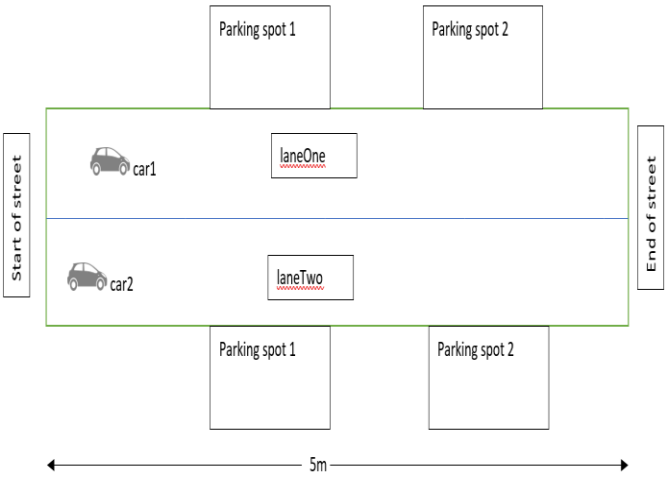


Figure 1. System state diagram.

### B. Uppaal

We used the Uppaal model checker which is an integrated tool environment that gives us the ability to model and verify the behaviour of a system [12]. Uppaal is a very powerful tool since it can handle real time issues and transitions. Bounded liveness can be expressed and verified in Uppaal which is something that many model checkers do not provide. Also, Uppaal has an easy to use interface which makes it easier to model and execute a system.

Moreover, Uppaal displays the sequence diagram of the system as it is being executed so every state can be checked and the user can see of the states are being changed in the same sequence as intended. There are also a lot of projects which have been tested by Uppaal which further provided me with the confidence to use it as the model checker and verifier for this project [13].

### C. Timed Automata

The Uppaal model checker was used to develop and test the model of a car which enters the parking system. As shown in Fig. 2, a car has four possible states, start, find_parking_spot, park and exit_street. A car can alternate between these four states.

For a car to transition from one state to another it must first fulfill the transition condition. As the car transitions from one state to another the position of the car which is stored in the car_pos variable is incremented. The car_pos variable helps detect the end of street.

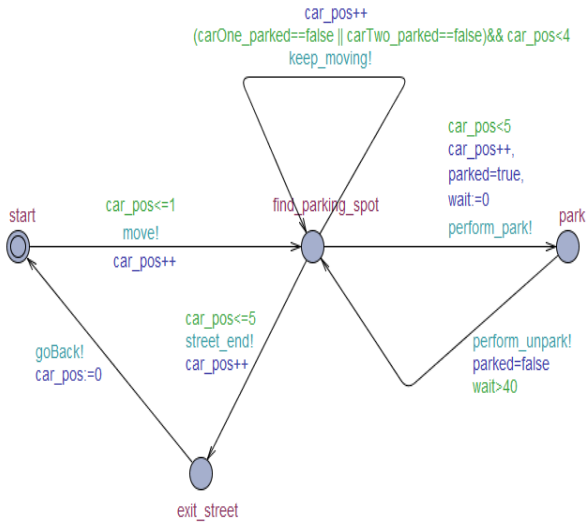


Figure 2. Model for car.

We also created a model for the lanes on which the cars can park. Each lane will have two parking spots as shown in Fig. 2. There are eight possible states in a lane. Similar to Fig. 1, the transition condition must first be fulfilled to transition from one state to another. A street_len variable is incremented whenever a car moves through the street. The street length is 5 meters.

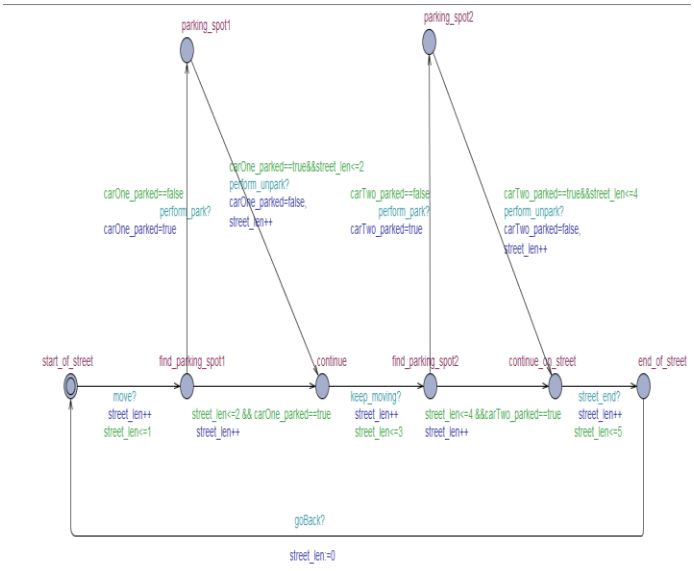


Figure 3. Model for lane.

The sequence diagram displayed in Fig. 4 shows the order in which states are executed and what triggers them. The states and transitions of the whole system are also displayed.

The sequence diagram can also be used to check the correctness of the system. In case of a deadlock, we can track the point at which deadlock occurs and can therefore, correct it.

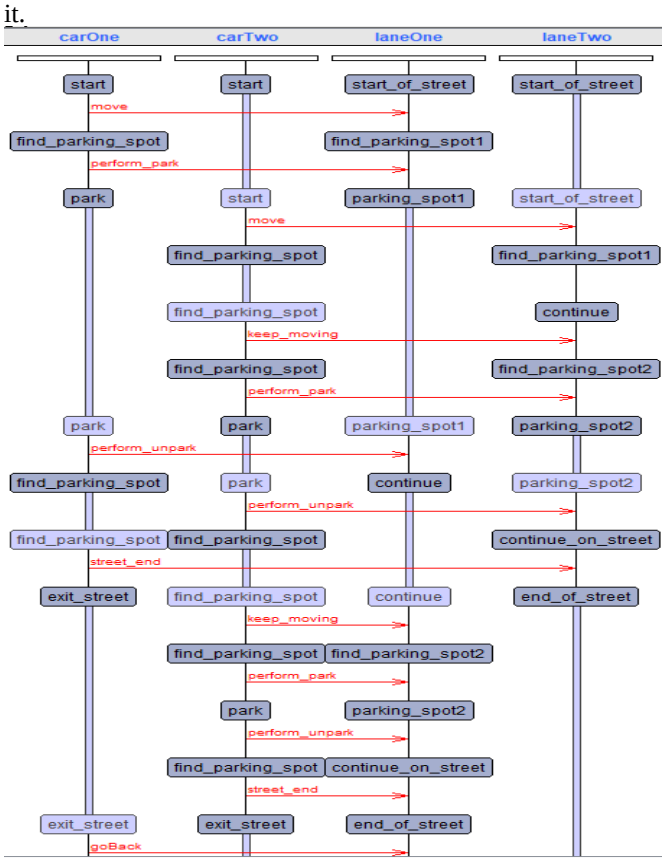


Figure 4. Sequence diagram

## IV. System Verification

The system was verified using the Uppaal model checker to ensure the safety, reachability, liveness, utility, deadlock freeness and fairness of the system.

### A. *Deadlock freeness*

- There are no deadlocks in the system at all.
  A[] not deadlock
- There are no deadlocks when the cars reach the end of the street.
  A[](deadlock imply (laneOne.end_of_street and laneTwo.end_of_street))

### B. *Safety*

- Car one cannot take up 2 parking spaces in lane one as it would cause no other car to park in lane one and if a car tries to park in lane one it would crash into car one.
  A[]((laneOne.parking_spot1 and carOne.parked==true)imply not(laneOne.parking_spot2 and carOne.parked==true))
- Car one cannot take up 2 parking spaces in lane two as it would cause no other car to park in lane two and if a car tries to park in lane two it would crash into car one.
  A[]((laneTwo.parking_spot1 and carOne.parked==true)imply not(laneTwo.parking_spot2 and carOne.parked==true))
- Car two cannot take up 2 parking spaces in lane one as it would cause no other car to park in lane one and if a car tries to park in lane one it would crash into car two.
  A[]((laneOne.parking_spot1 and carTwo.parked==true)imply not(laneOne.parking_spot2 and carTwo.parked==true))
- Car two cannot take up 2 parking spaces in lane two as it would cause no other car to park in lane two and if a car tries to park in lane two it would crash into car two.
  A[]((laneTwo.parking_spot1 and carTwo.parked==true)imply not(laneTwo.parking_spot2 and carTwo.parked==true))

### C. *Reachability*

- Car one exits the street infinitely often.
  E<> (carOne.exit_street)
- Car two exits the street eventually.
  E<> (carTwo.exit_street)

### D. *Liveness*

- Car one or car two or both must always be looking for a parking space for the system to stay alive.
  E<> ((carOne.find_parking_spot and carTwo.park) or (carTwo.find_parking_spot and carOne.park) or (carOne.find_parking_spot and carTwo.find_parking_spot))

### E. *Mutual exclusion*

- Car one cannot be parked in lane one and lane two at the same time.
  A[]((laneOne.parking_spot1 and carOne.parked==true)imply not(laneTwo.parking_spot1 and carOne.parked==true))
- Car two cannot be parked in lane one and lane two at the same time.
  A[]((laneOne.parking_spot1 and carTwo.parked==true)imply not(laneTwo.parking_spot1 and carTwo.parked==true))
- Car one cannot be at the start and the end of the street at the same time.
  A[]((carOne.start and laneOne.start_of_street)imply not(carOne.start and laneOne.end_of_street))

 ISBN: 978-1-61208-752-8 

- Car two cannot be at the start and the end of the street at the same time.
  A[](((carTwo.start and laneOne.start_of_street)imply not(carTwo.start and laneOne.end_of_street))

### F. Utility

- If carOne enters the street, it eventually parks on the street.
  E<> (carOne.start imply carOne.park)
- If carTwo enters the street then it eventually parks on the street.
  E<> (carTwo.start imply carOne.park)

### G. Fairness

- Neither car one nor car two waits for longer than 40 seconds to unpark.
  A<> (carOne.wait<=40 and carTwo.wait<=40)
- Cars cannot drive on the street for longer than the length of the street.
  A[] (laneOne.street_len<6 and laneTwo.street_len<6)

Some of the verified properties are displayed in Fig. 5.

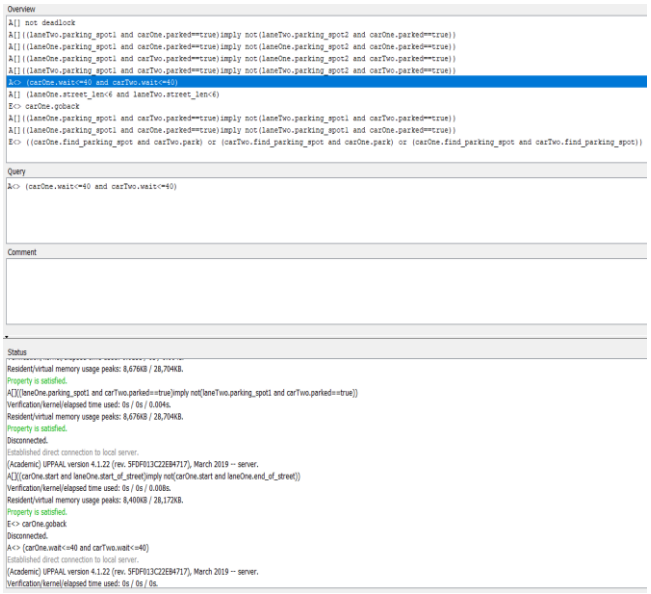


Figure 5. Results of the uppaal model checker

## V. Conclusion and Future Work

In this paper, an effective methodology for the modeling and verification of the car parking system was shown. The safety properties were verified using Uppaal and the stability of the real time automata was checked through the model checker as well. The car and lane models communicate effectively through common channel.

Most of the existing smart car parking systems are tested using hardware simulations which do not always ensure the correctness of the system. On the other hand, we have made use of formal modeling and verification techniques to prove that our model is correct.

For the time being we have developed a simple car parking system which is implemented in the real world scenario using the Uppaal model checker. To further improve our work, the system can be extended to accommodate a greater number of cars, lanes and parking spots. Moreover, a module can be added to find out and direct the car to the closest parking spot.

## References

[1] M. Jaffar-ur Rehman, F. Jabeen, A. Bertolino and A. Polini, "Testing software components for integration: a survey of issues and techniques", Software Testing, Verification and Reliability, vol. 17, no. 2, pp. 95-133, 2007.

[2] P. Upadhyay, "The Role of Verification and Validation in System Development Life Cycle", IOSR Journal of Computer Engineering, vol. 5, no. 1, pp. 17-20, 2012.

[3] M. Latuszynska, “Problems of verification and validation of computer simulation models”, STUDIA INFORMATICA, pp. 27-40, 2013.

[4] G. Behrmann, A. David, K. Larsen, P. Pettersson and W. Yi, "Developing UPPAAL over 15 years", Software: Practice and Experience, vol. 41, no. 2, pp. 133-142, 2011.

[5] M. Ahmed and W. G. Wei, (2014). “Study on Automated Car Parking System Based on Microcontroller”, International Journal of Engineering Research & Technology, vol. 3, no. 3, pp. 256-258, January 2014.

[6] S. Ghosh, S. Prusty and P. B. Natarajan, “Design and Implementation of Smart Car Parking System Using LabVIEW”, International Journal of Pure and Applied Mathematics, vol. 120, pp. 329-338, October 2018.

[7] M. Sabnam, M. Das, P. A. Kashyap, “Automatic Car Parking System”, ADBU Journal of Engineering Technology, vol. 4, no. 1, 2016.

[8] A. Ahad, Z. Khan, and S. Ahmad, “Intelligent Parking System”, World Journal of Engineering and Technology, vol. 4, no. 2, pp. 160-167, May 2016.

[9] J. D. Bachhav1, Mechkul, “Smart Car Parking System”, International Research Journal of Engineering and Technology (IRJET), vol. 4, no. 6, pp. 3036-3038, June 2017.

[10] S. Latif, H. Afzaal and N. A. Zafar, "Modelling of Graph-Based Smart Parking System Using Internet of Things", International Conference on Frontiers of Information Technology (FIT), pp. 7-12, 2018.

[11] “Formal Methods in the Teaching Lab Examples, Cases, Assignments and Projects Enhancing Formal Methods Education”, Formal Methods Europe Subgroup on Education, Hamilton, ON, Canada, 2006, pp. 61-62

[12] M. P. Júnior and G. V. Alves, “A Study Towards the Application of UPPAAL Model Checker”, 3rd Workshop-School on Theoretical Computer Science, pp. 5-8, September 2015.

[13] A. Hessel, K. Larsen, M. Mikučionis, B. Nielsen, P. Pettersson and A. Skou, “Testing real-time systems using UPPAAL”, Formal Methods and Testing. pp. 77-117, January 2018.

 ISBN: 978-1-61208-752-8 