



# Logic in Computer Science

## Assignment

**Group Project on Modelling Traffic System in UPPAAL**

**Submitted to**

**Dr. Jagat Sesh Challa**

### **Assignment Description:**

The assignment focuses on the design and verification of a Smart Intersection system using the UPPAAL model checker. The work involves modelling traffic lights, pedestrian crossings, and emergency vehicle priority with realistic timing constraints and coordinated system behavior. The aim is to capture complex intersection dynamics through timed automata and to evaluate the system through simulation under normal traffic, pedestrian activity, and emergency scenarios. The assignment also includes the verification of key safety and liveness properties to ensure that the intersection operates correctly and efficiently in all considered conditions.

**Submitted by**

**Dhaval Bothra, ID : 2024A7PS0536P**

**Kandarp Kalavadiya, ID : 2024A7PS0582P**

**Rachit Soni, ID : 2024A7PS0587P**

**Ujwal Keshava, ID : 2024A7PS0608P**

**Girish Mundra, ID : 2024A7PS0993P**

NOVEMBER 2025

## ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **Prof. Jagat Sesh Challa** for providing us with a strong foundation in Propositional Logic, Predicate Logic, and Floyd Hoare Logic. His clear explanations and structured approach greatly shaped our understanding of formal methods, which became essential while working on this assignment.

We are equally grateful to **Prof. Rajesh Kumar** for his engaging and insightful lectures on Satisfiability Solvers, Linear Temporal Logic and Computation Tree Logic. His emphasis on temporal reasoning and system verification helped us appreciate the theoretical depth behind the properties we analysed and modelled in UPPAAL.

We would also like to extend our appreciation to **Teaching Assistant Shivam Goyal**, whose tutorial sessions on UPPAAL were instrumental in helping us navigate the tool effectively. His patient guidance and hands-on demonstrations made the modelling and verification process far more approachable.

This assignment provided us with an opportunity to connect theoretical concepts with practical system modelling, and the collective support from our instructors made the experience both enriching and rewarding. We are truly thankful for their guidance throughout the course.

# 1 Introduction

Modern traffic intersections involve complex coordination among vehicles, pedestrians, and emergency responders. Traditional fixed-timer traffic controllers are insufficient in dynamic and high-traffic environments. Therefore, this project models a **Smart Intersection System** using the UPPAAL model checker, leveraging timed automata to simulate real-time constraints, synchronization between subsystems, and safety requirements.

The objectives of the project are:

- To model vehicles, pedestrian behaviour, and emergency priority within an intersection.
- To design templates representing traffic lights, controllers, arrival processes, and crossing logic.
- To ensure correctness through simulation and temporal-logic verification.
- To verify key safety, reachability, and liveness properties.

This system models realistic constraints such as minimum/maximum green periods, pedestrian crossing time, emergency handling, and queue reduction. The following sections describe each template in detail.

## 2 Assumptions

The model incorporates the following assumptions for simplifying and formalizing the intersection behaviour:

1. **Intersection Structure** Two main traffic directions: North–South ( $\text{dir} = 0$ ) and East–West ( $\text{dir} = 1$ ). Four sub-directions for arrivals: 0, 1, 2, 3, where South–North is  $\text{dir} = 2$  and West–East is  $\text{dir} = 3$ .
2. **Timing Constraints**
  - Maximum Green: 60 seconds
  - Minimum Green: 15 seconds
  - Yellow Time: 3 seconds
  - All-Red Buffer: 1 second
  - Vehicle/Pedestrian Crossing Time: 3 seconds
3. **Queue Behaviour** Up to 60 vehicles/pedestrians may cross per activation cycle.
4. **Emergency Vehicle Handling** Emergency arrivals override normal behaviour only after the current direction has crossed its minimum green time. Also we assume while crossing that the emergency vehicle are at the front-most location.

5. **Pedestrian Safety** Pedestrian walk signal is never active when vehicles have green in the same direction.
6. **Channel Communication Assumption** All UPPAAL synchronisations occur without delay or loss.
7. **Deterministic Model** Arrival rates are nondeterministic but not probabilistic. The system behaviour is deterministic under UPPAAL semantics.

### 3 Template Descriptions

This section describes each UPPAAL template used in the model, along with an image placeholder for the corresponding automaton diagram.

#### 3.1 CentralController Template

The CentralController coordinates the global traffic phases. It ensures that:

- North–South and East–West directions never receive green simultaneously.
- Minimum and maximum green times are enforced.
- Emergency vehicle arrival triggers yellow transitions.
- All-red safety buffers are satisfied before direction switching.

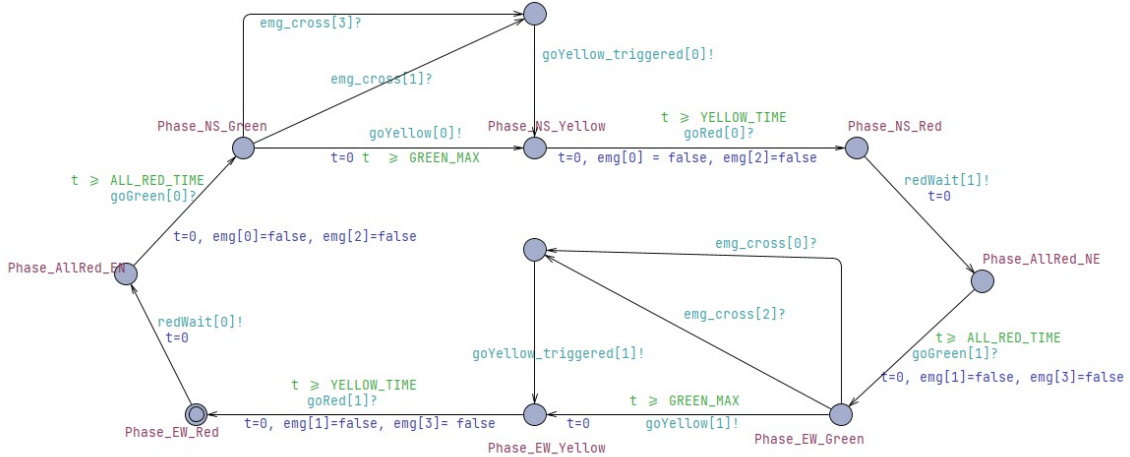


Figure 1: CentralController Automaton

### 3.2 TrafficLight(dir) Template

This template governs the traffic signal for each direction. It includes four states: *Green*, *Yellow*, *RedBuffer*, and *Red*. It synchronizes with the CentralController using:

- goGreen[dir]!
- goYellow[dir]? and goYellow\_triggered[dir]?
- goRed[dir]!
- redWait[dir]?

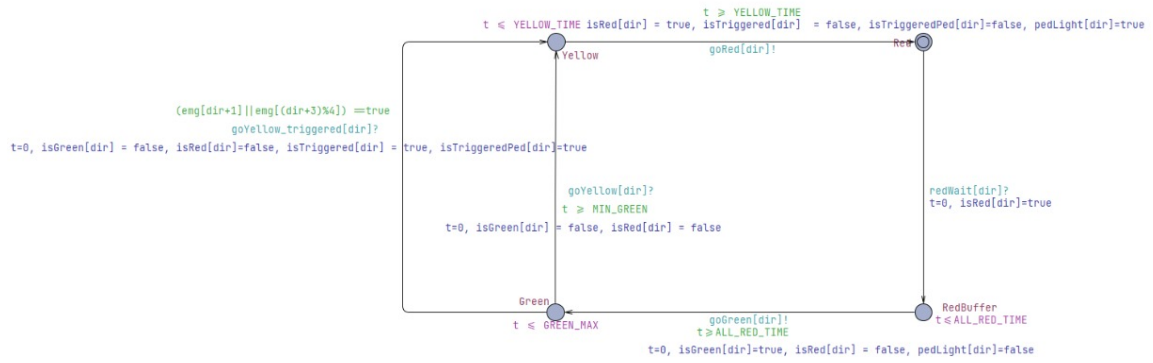


Figure 2: TrafficLight Template

### 3.3 VehicleArrival(dir) Template

This template simulates nondeterministic arrival of vehicles. Each arrival increments the queue:

$$vehQ[dir] \leftarrow vehQ[dir] + 1$$

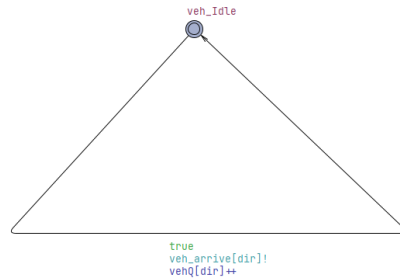


Figure 3: VehicleArrival Template

### 3.4 VehicleCrossing(dir) Template

Responsible for the movement of vehicles once the direction is green. It:

- Checks whether `isGreen[dir%2]` is true.
- Triggers `start_cross[dir%2]!`.
- Removes up to 60 vehicles from the queue.

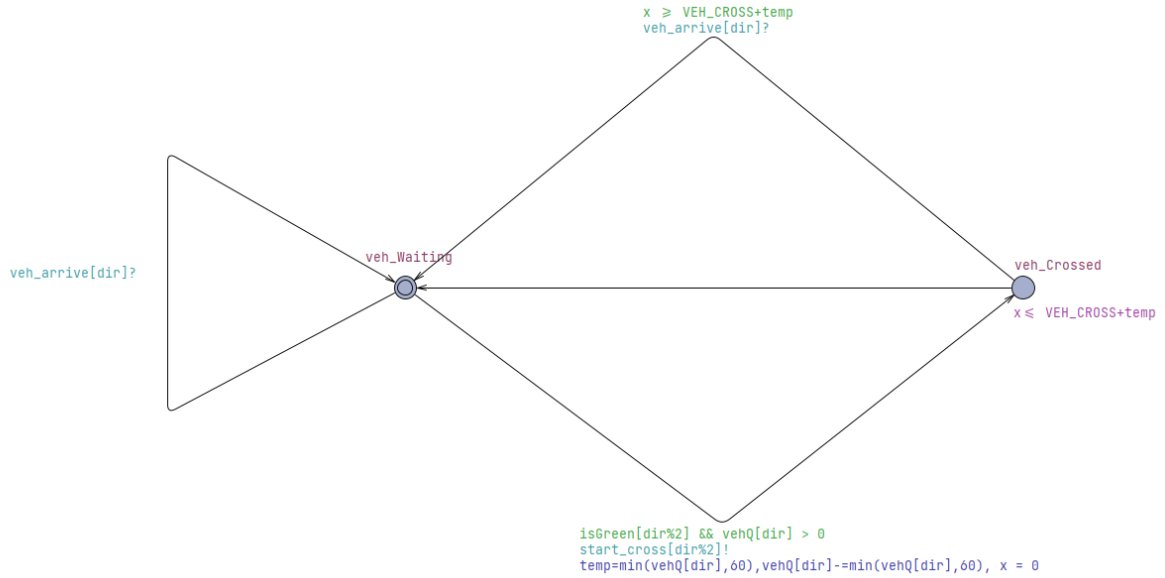


Figure 4: VehicleCrossing Template

### 3.5 Ped\_Arrival(dir) Template

Models pedestrian arrival using:

$$pedQ[dir] \leftarrow pedQ[dir] + 1$$

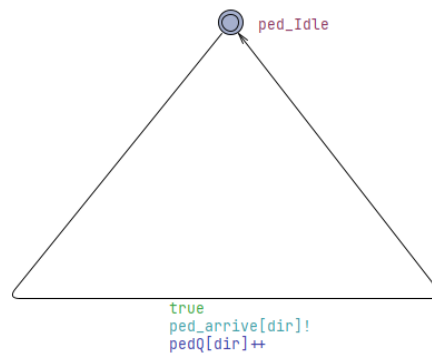


Figure 5: Ped\_Arrival Template

### 3.6 Ped\_Controller(dir) Template

Controls pedestrian crossing behaviour. It ensures:

- Pedestrians cross only when the corresponding direction is red.
- A batch of pedestrians (up to 60) cross safely.

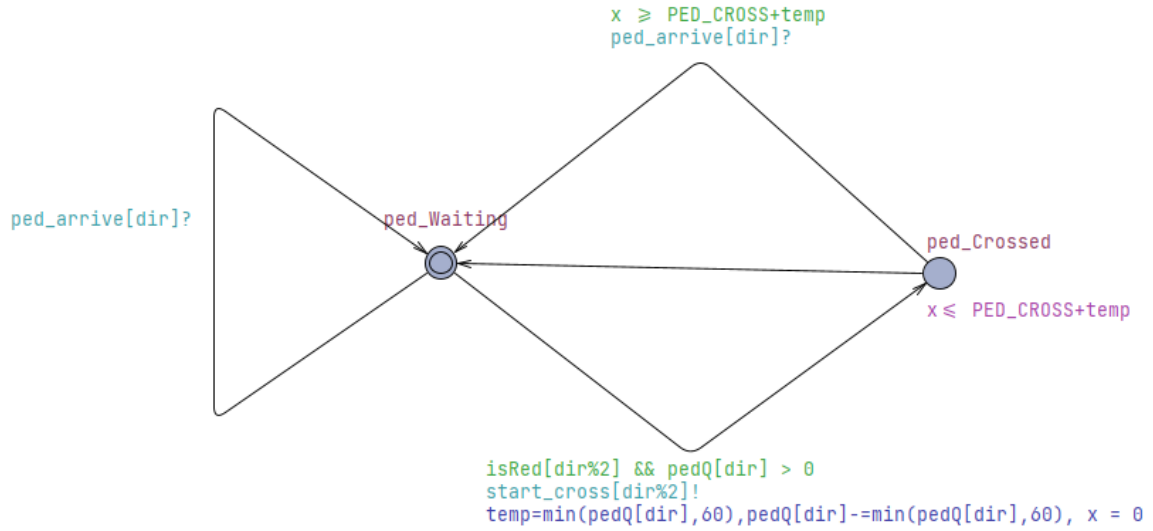


Figure 6: Ped\_Controller Template

### 3.7 Emg\_Arrival(dir) Template

This template generates emergency vehicle arrival signals.

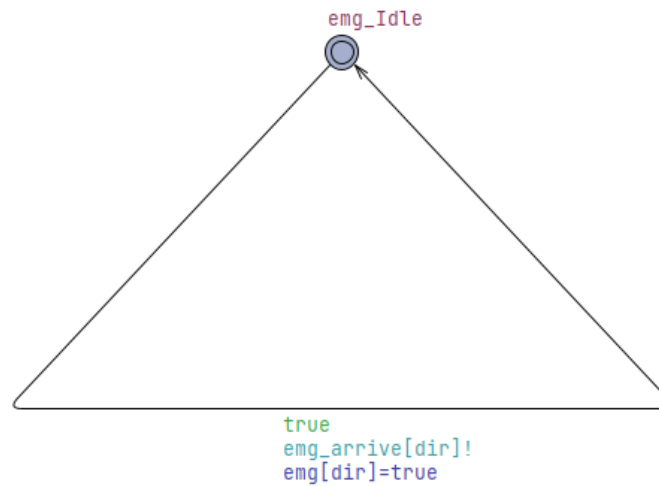


Figure 7: Emg\_Arrival Template

### 3.8 Emg\_Controller(dir) Template

Handles emergency priority. If an emergency arrives and the direction is not green, it sends:

$emg\_cross[dir]!$

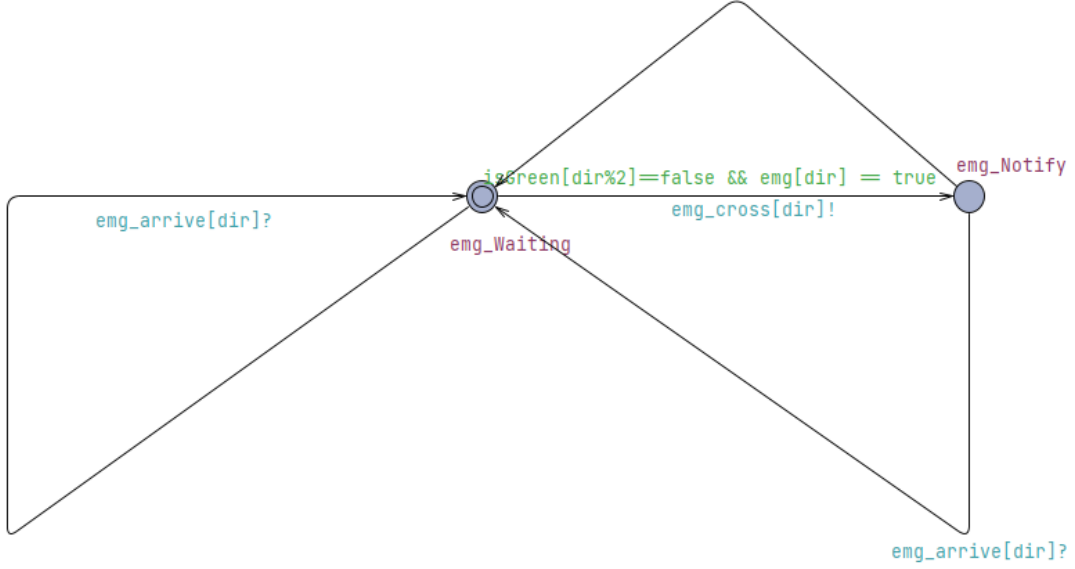


Figure 8: Emg\_Controller Template

## 4 Verification of System Properties (IMMEDIATE Policy)

For this project we implemented the **IMMEDIATE** emergency-preemption policy: an emergency arrival can pre-empt ongoing normal operation (after minimum safety constraints) to give immediate priority to the emergency vehicle's direction.

During verification we attempted to check both **safety** and **liveness** properties using UPPAAL's exhaustive model checker. However, several universal safety properties expressed with the **A[]** operator caused **state-space explosion** in the model: the number of reachable states grew beyond what our machine (and UPPAAL's exhaustive engine) could handle within acceptable time/memory limits.

Below we state the properties we intended to verify, what we actually verified, and the strategies we applied to obtain useful results despite the state-space explosion.

### 4.1 Intended Properties (Formalised)

**Safety (intended)**

1. No simultaneous vehicle greens (NS vs EW):

$$A[] \neg(isGreen[0] \wedge isGreen[1])$$

2. No pedestrian walk while conflicting vehicle green:

$$A[] \neg(pedLight[i] \wedge isGreen[i]) \quad (i = 0, 1)$$

3. Emergency preemption preserves safety (mutual exclusion):

$$A[] \neg(isGreen[0] \wedge isGreen[1])$$

### Liveness (intended)

$$A\Diamond(vehQ[i] == 0) \quad \text{and} \quad A\Diamond(pedQ[i] == 0) \quad (i = 0..3)$$

### IMMEDIATE policy properties

1. Reachability of preemption (interruption possible):

$$E\Diamond (isTriggered[i] \wedge isTriggeredPed[i])$$

2. Emergency receives green within 5 seconds (bounded response):

$$A[] (emg[dir] \Rightarrow \Diamond_{\leq 5} isGreen[dir])$$

## 5 Group Member Contributions

Although everyone had significant contributions in each component, these are the major contributions.

| Member             | Major Contribution   |
|--------------------|--|
| Dhaval Bothra      | Modelling Vehicle logic and Central Controller templates.      |
| Kandarp Kalavadiya | Modelling Pedestrian logic templates and document preparation. |
| Rachit Soni        | Modelling Traffic Light and Emergency vehicle logic templates. |
| Ujwal Keshava      | Modelling Traffic Light and Emergency vehicle logic templates. |
| Girish Mundra      | Modelling Vehicle logic and Central Controller templates.      |

## 6 References

### References

- [1] Behrmann, G., David, A., & Larsen, K. G. (2004). *A Tutorial on UPPAAL*. Aalborg University. Available at: <https://homes.cs.aau.dk/~bnielsen/MTV08/material/uppaal-tutorial.pdf>
- [2] UPPAAL Documentation — Concrete Simulator. *Available at:* <https://docs.uppaal.org/gui-reference/concrete-simulator/>
- [3] KF6009 Traffic Light Control System (GitHub Repository). *Available at:* <https://github.com/danielkelshaw/KF6009-TrafficLight>
- [4] UPPAAL Documentation — Expressions and Language Reference. *Available at:* <https://docs.uppaal.org/language-reference/expressions/>
- [5] Thamilselvam, B., & Kalyanasundaram, S. *Coordinated Intelligent Traffic Lights using UPPAAL Stratego*. Available at: <https://people.iith.ac.in/subruk/pdf/uppaal.pdf>
- [6] Larsen, K. G., Pettersson, P., & Yi, W. (1997). *UPPAAL in a Nutshell*. In Proceedings of the REX Workshop, LNCS 600, Springer. Available at: <https://uppaal.org/texts/lpw-sttt97.pdf>