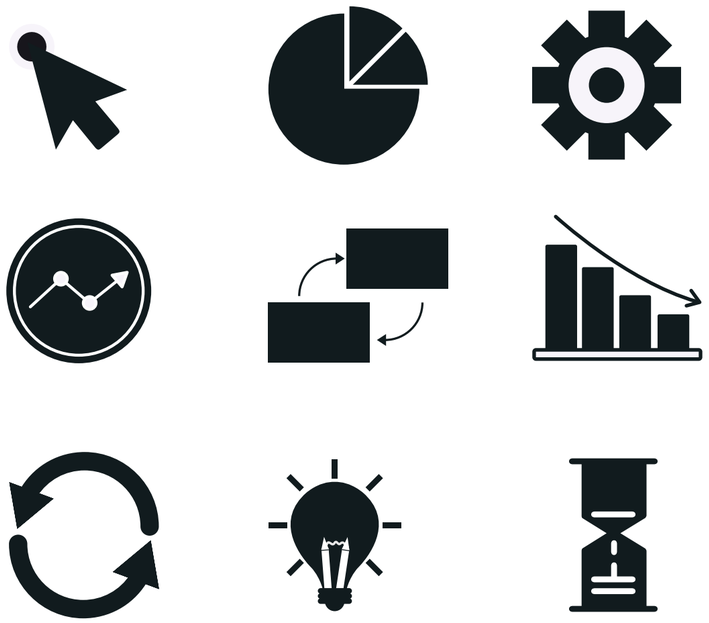
APRIL. 27, 2020 • NCSU

Cyber Physical Systems

Uppaal

Important concepts



Cyber Physical Systems

CPS AND UPPAAL

Allowing a imaginable user interface, Uppaal make it easy to create a notion of the system in mind. It facilitates the validation of the system by adding an all time verification process to check if any condition is true by all possible routes of execution

TIMING

A constantly varying quantity that time is, is handled with detailed precision in Uppaal through the concrete simulator which allows the user to decide every aspect of timing (like delay) in the process.

CHANNEL PRIORITIES

Prioritizing a channel over another when there is a possibility for both to synchronize is very essential in some systems. The 'channel priority' reduces the choices the user has to make by not showing the channel with low priority as an option.

QUERIES

Verification divided into Reachability, Safety, Liveness, and Deadlock is very necessary to understand the behaviour of the system in any possible way. It also allows designing the system in a way we want it to work

DISCUSSION OUTLINE

Problems

Bridge
Online Booking
Butler
Museum
Roller Coaster
Parcel Router
Elevator
Freeway

Bridge

PROBLEM

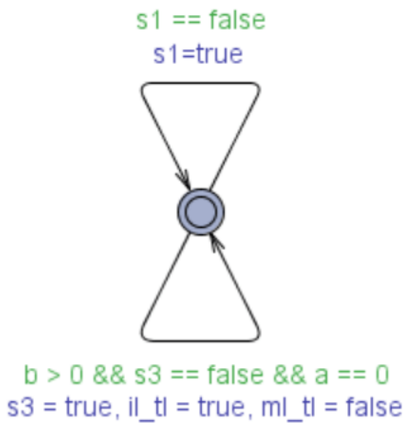
The problem has a one-way bridge that connects an island to the mainland. Number of the cars on the mainland is not certain however there is a limit for number of cars on bridge and island. Two traffic lights and four sensors are located entrance of bridge, mainland and island in two ways. While sensors giving information about the position of cars, traffic lights change their colors to control the traffic on the bridge.

STRATEGY

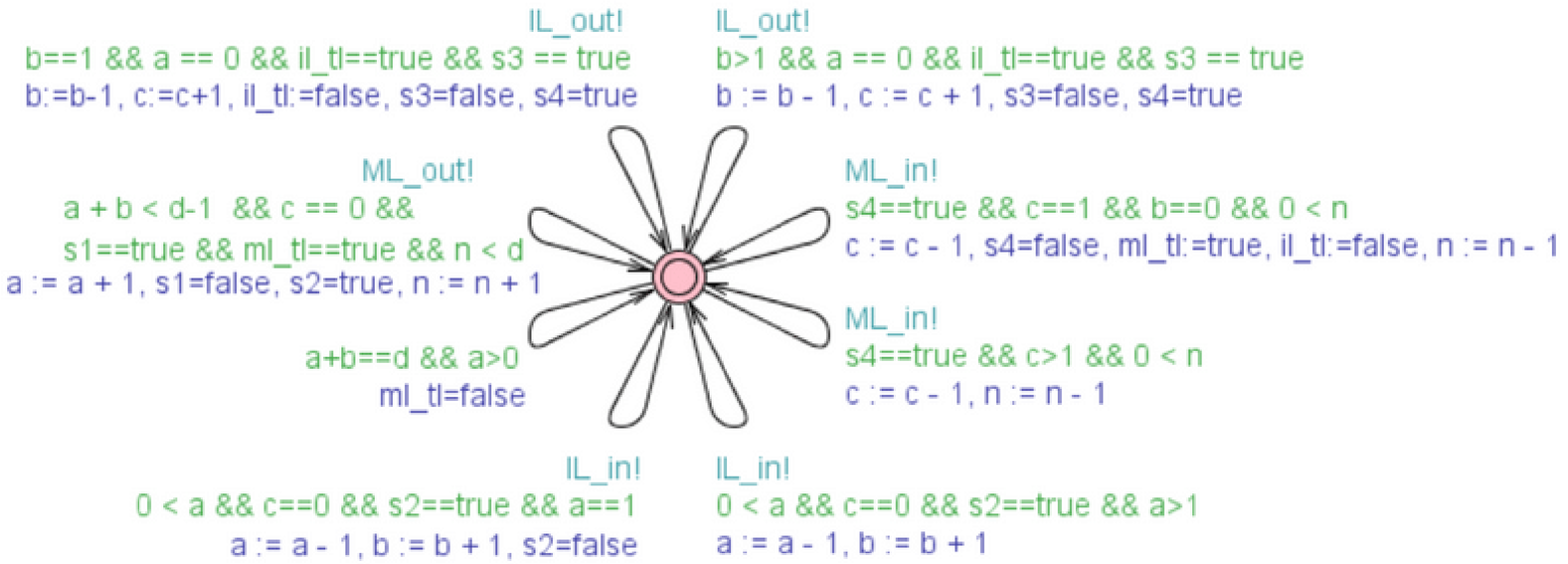
The passing of the vehicles is controlled by observing the status of the traffic lights updated by reading the sensor data. For all possible transitions, the limit of the number of cars that can be found on the bridge and the island took into account.

MODEL

The plant includes main movements ML_out(leaving the mainland), ML_in(entering to the mainland). Similarly, island transactions are modeled(IL_in, IL_out). Number of cars are declared as different counters a(cars going to island from the mainland), b(cars on the island), c(cars going to the mainland from island).

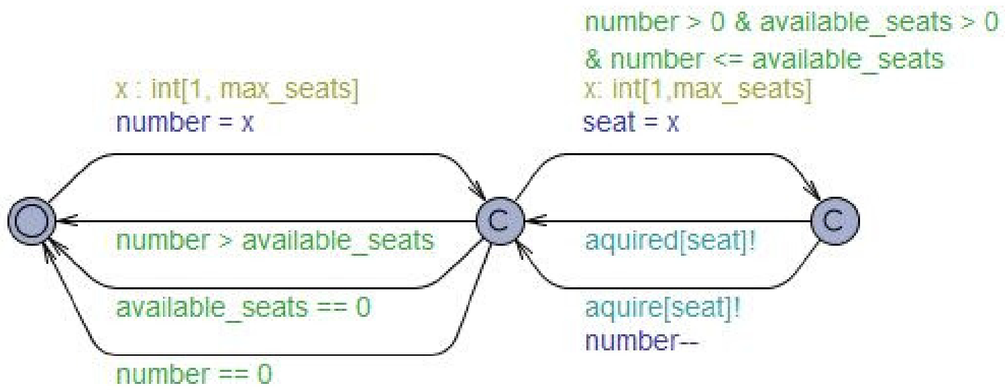


CONTROLLER

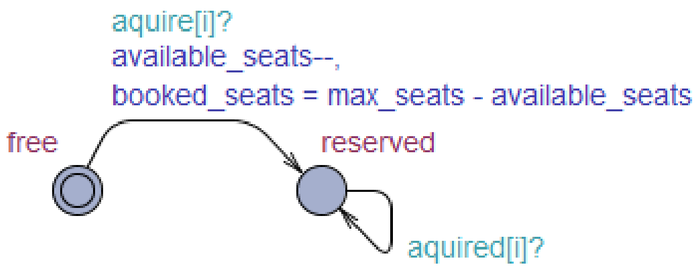


PLANT

Online Booking



BROWSER



SEAT

PROBLEM

A web-based application to reserve seats for movies is designed. The current state of reservations is viewed on the browser. The user chooses the number of seats and location of them. If the seats selected are available, the system blocks them.

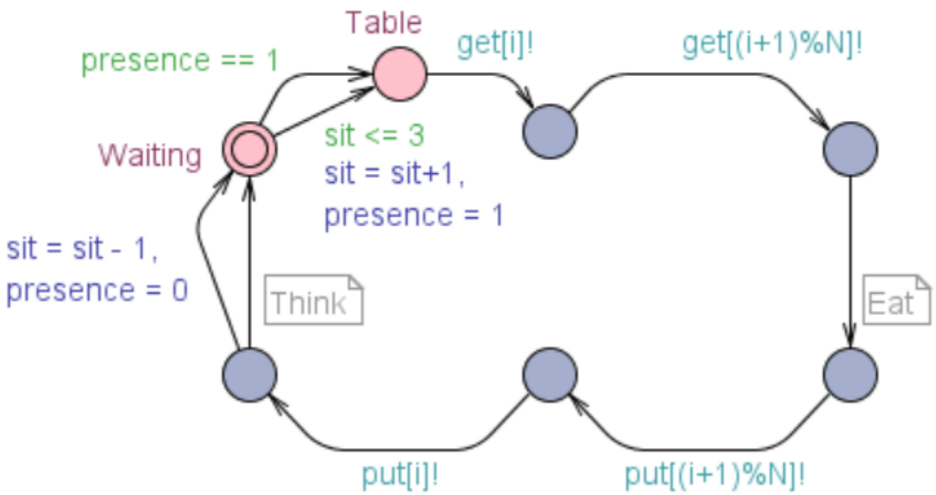
STRATEGY

To allow multiple seat booking, a choice for number of seats is given before booking the seats. A check for number of seats being less that seats available once satisfies, the user is allowed to select the seats, thus acquiring them. The seat is shows as a template instead of an integer to allow the visibility of booked seats.

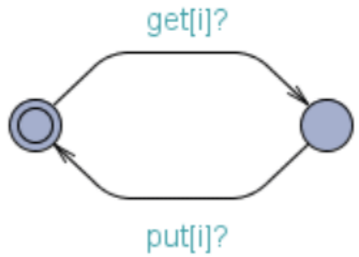
MODEL

The Browser template synchronizes with the Seat template to acquire it. The model only proceeds to book the seats if number of seats requested is less that the number of seats available. A seat is only acquired if it isn't yet. 'select' is used for number of seats as well as seat selection for randomness

Butler



PHILOSOPHERS



FORKS

PROBLEM

Five philosophers who spend their time alternately eating and thinking share spaghetti from the plate in the middle of the round table. Although each philosopher needs two forks to eat, there are only five forks on the table. The butler problem, which is a solution to the problem of dining philosophers, regulates the eating process by allowing only four philosophers to sit at the table at a time.

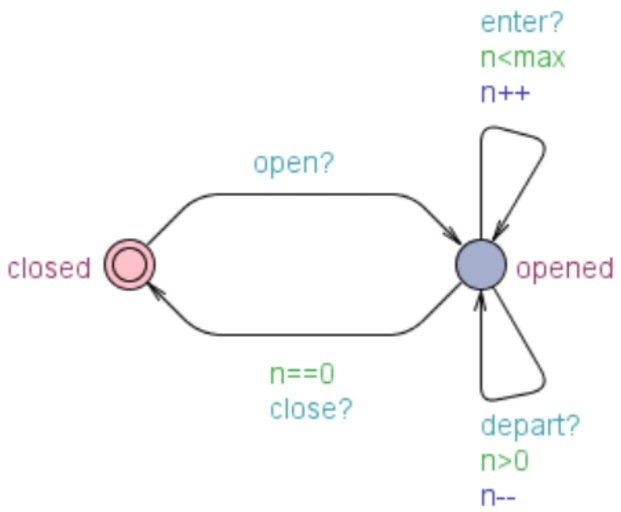
STRATEGY

After checking for number of philosophers sitting is less than or equal to three any philosopher is allowed to sit and eat. Otherwise they need to wait for the next empty spot. The 'sit' variable is restricted to not go above four to allow continuous eating of philosophers.

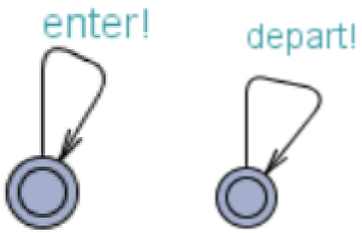
MODEL

The 'philosophers' template synchronizes with 'forks' template when philosophers grab the forks before eating. After eating, philosophers can think and wait or they can choose to stay at the table. Passing 'i' as a parameter allow us to see the state of each philosopher and fork and using modulo (%) expression to assign their place at the table.

Museum



CONTROLLER



GATES

DIRECTOR

PROBLEM

Museum with two gates (East for entrance, West for departure) has a director who decides opening and closing state of the museum. Controller for the gates collects signal from director and turnstiles located at gates to prevent exceeding limit for number of visitors.

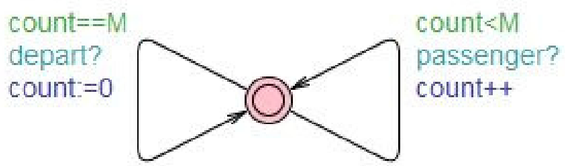
STRATEGY

Entrances and departures of visitors are handled in opened state of museum model by synchronized channels with turnstiles. Visitors are limited by variable 'max' and when there is no visitor inside, director machine can work with controller to close the museum. East and West gates creates random visitor behaviour for the system.

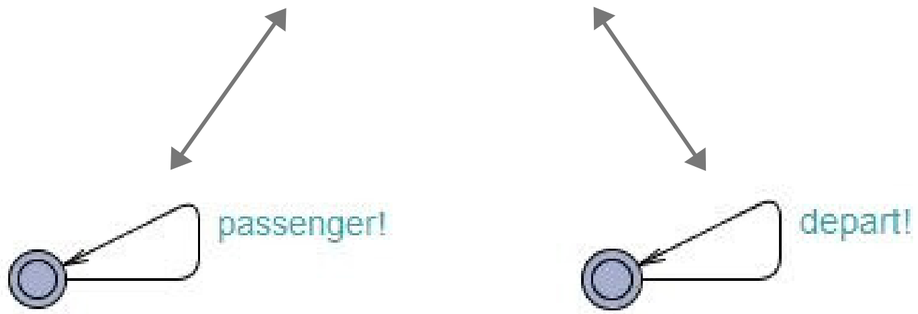
MODEL

The model includes a controller, two gates, and a director. The controller synchronizes with the gates and director machine by enter!, depart!, open! and close! channels

Roller Coaster



CONTROLLER



TURNSTILE

CAR

PROBLEM

A roller coaster system is a car which provides a thrilling ride to the passengers. Passengers arrive at the ride and register with a turnstile. Once the ride is full, passengers are allowed to occupy the seat and enjoy the ride. Once the ride is complete, car is emptied and a new set of passengers are allowed to ride.

STRATEGY

This toy problem only considers the capacity of the car as a restriction. Thus, once the limit is reached, turnstile is locked and people behind wait for the next ride. A random people generator edge is used to show the passengers.

MODEL

The model consists of three templates, a controller, turnstile, and a car. The controller synchronizes with the turnstile and car with channels passengers! and depart!

Parcel Router

PROBLEM

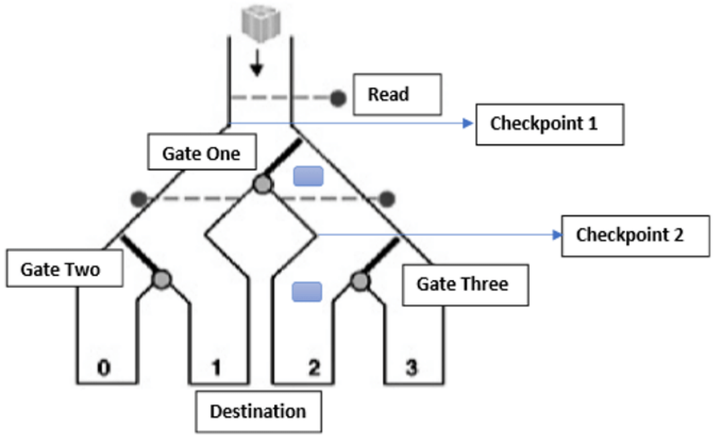
Parcels drop from the top of the system into chutes and they move along by gravity. Every parcel has a barcode between 0 to 3 that shows their destination. When parcel enters the system sensor reads the barcode to send parcels to the right destinations. Gates should be controlled for creating pathways for each parcel and they can be moved when there is no parcel in the way.

STRATEGY

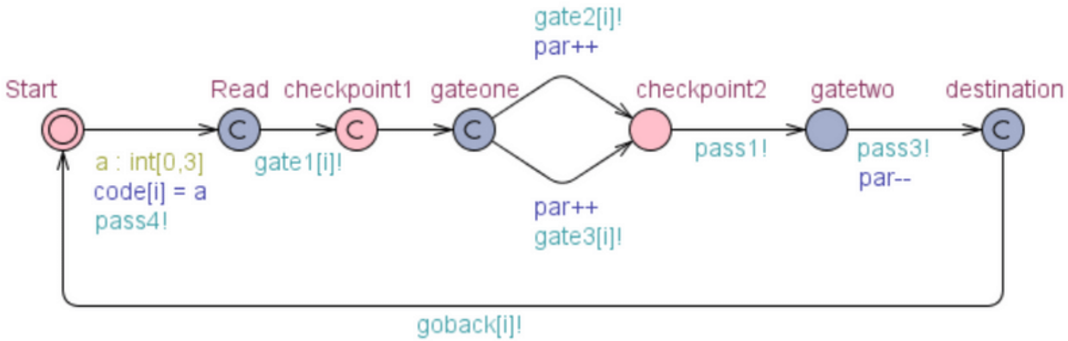
Sensors are located at checkpoints. A new parcel is added when the previous parcel passes gate one. Movements of the parcels that follow each other modeled in a pattern allowing them to move accordingly. Thus, creating a system with a maximum of three parcels at the same time.

MODEL

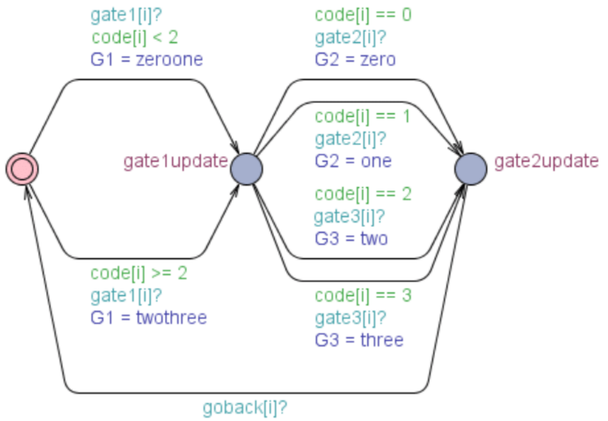
Parcel router model has three plants representing three packages in the system and a machine named 'Gates' that works simultaneously with the plants using synchronized channels. Checkpoints are positioned just before the gates, and we update gate switches before checkpoints to make sure that gates are at their correct position before the parcel in the way.



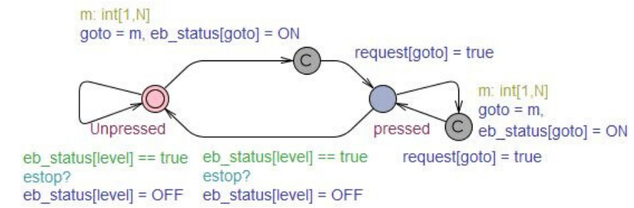
SYSTEM



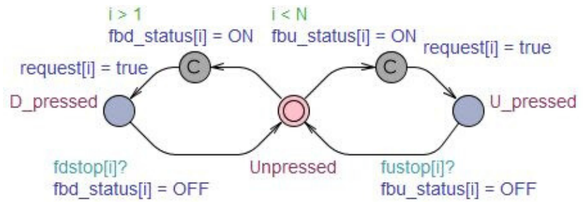
PLANT



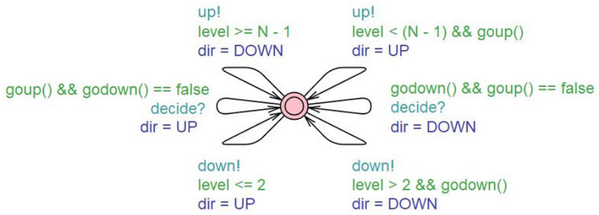
GATES



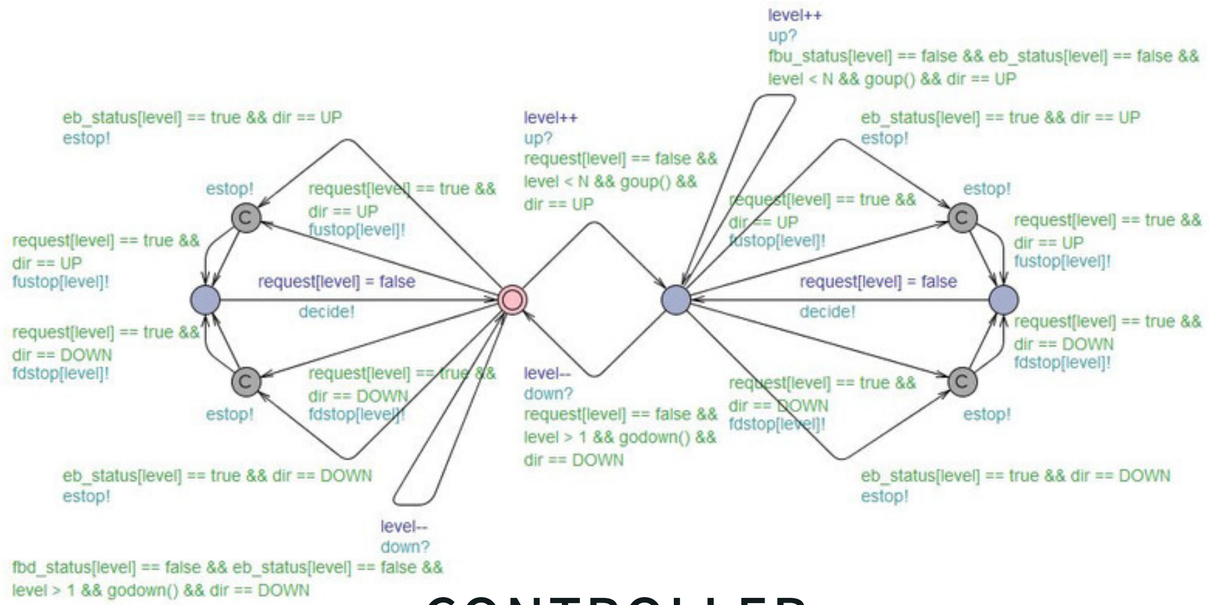
ELEVATOR BUTTON



FLOOR BUTTON



SERVER



CONTROLLER

Elevator

PROBLEM

An elevator simulator is designed to control an elevator in a building with N floors. Buttons on floor, and in the elevator are responsible for updating the status of request in the server which also decides the trip in which the request is to be satisfied. A controller is used to operate the elevator.

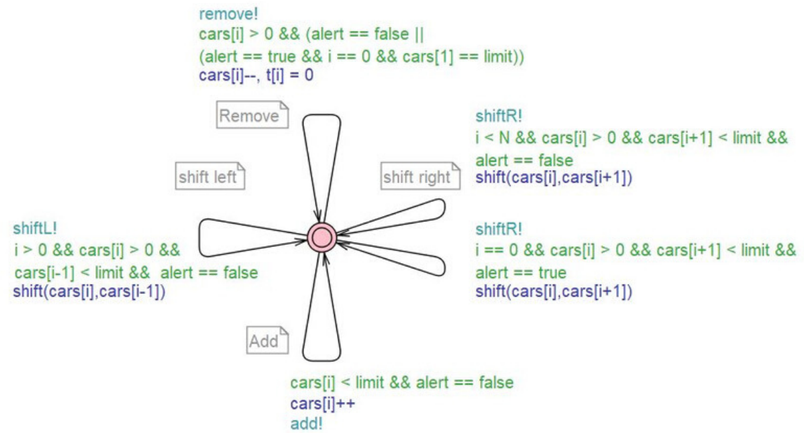
STRATEGY

The elevator is designed to go to the floor it has a request for. If it does not have any request, it stays at the floor it is at. Preference is to the direction the elevator is going to (up or down). Only if there isn't a request in the current direction, the elevator goes the other way.

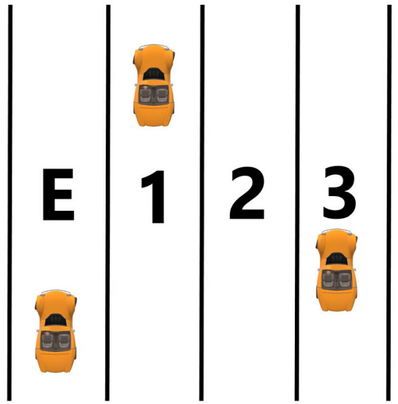
MODEL

The model is divided in three type of functions: buttons, server, and controller. The buttons are further divided as elevator and floor buttons. A request array is used to combine the requests by buttons. A function is defined to check if there is any request above the current level. The server decides the trip for each request. The controller has a setup where the elevator movement is based on all of the restrictions (as guards).

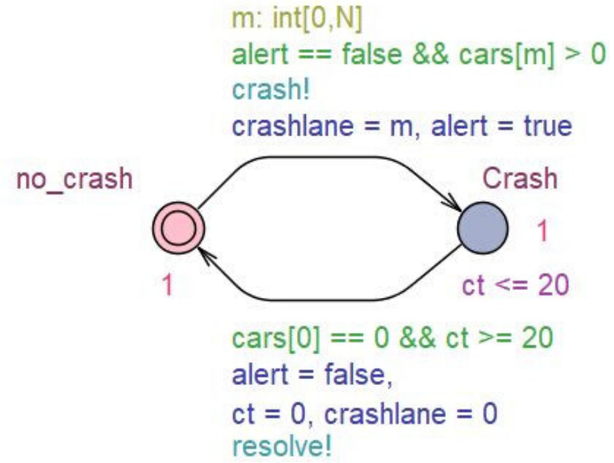
Freeway



LANE



FREEWAY



CRASH GENERATOR

PROBLEM

A freeway has three lanes and an emergency lane. Each lane has a car capacity of five. The emergency lane can be used as a normal lane during normal operations. In case of an accident, cars must move out of the emergency lane to allow the officials to take the lead. The goal is to create a dead-lock free model.

STRATEGY

To make sure that the emergency lane is empty in case of an emergency, we either move the cars to the first lane or if that is not possible, out from the emergency lane. In normal operations, cars can shift to both sides. For a crash in lane 1, an assumption that cars from the emergency lane shift to lane 1 ahead of the crash. Once there is a crash on the freeway, highest priority is to resolve it before the freeway is back to normal operations.

MODEL

The 'Lane' template does adding, removing and shifting of cars between lanes. A 'shift' function is responsible for removing a car from a lane and adding it to the one besides it. The 'crash generator' template is used to create an instance of an accident on any lane (selected randomly).

The Presenters

AKASH LODHA

GULSAH CAKIR

Cyber Physical Systems • Spring 2020