National UniversityA logo of a university

Description automatically generated

of Sciences and Technology (NUST)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Formal Methods

SE 320 Assignment 2

BESE 12-B  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Instructor : Dr. Sidra Sultana**

|  |  |
| --- | --- |
| **NAME** | **CMS** |
| Laiba Atiq | 372187 |
| Imaan Ibrar | 373982 |
| Abdullah Tahir | 393056 |

Contents

[Introduction: 3](#_Toc167006707)

[System Overview 3](#_Toc167006708)

[Timed Automata Modeling 4](#_Toc167006709)

[PARKING LOT ENTRY GATE TIMED AUTOMATA 4](#_Toc167006710)

[Locations: 5](#_Toc167006711)

[Clocks: 5](#_Toc167006712)

[Channels: 5](#_Toc167006713)

[Transitions: 5](#_Toc167006714)

[Invariants: 6](#_Toc167006715)

[PARKING LOT EXIT GATE TIMED AUTOMATA 6](#_Toc167006716)

[Locations: 6](#_Toc167006717)

[Clocks: 6](#_Toc167006718)

[Channels: 6](#_Toc167006719)

[Transitions: 7](#_Toc167006720)

[Invariants: 7](#_Toc167006721)

[CAR PARKING SPACE FINE TIMED AUTOMATA 7](#_Toc167006722)

[Locations: 8](#_Toc167006723)

[Clocks: 8](#_Toc167006724)

[Channels: 8](#_Toc167006725)

[Transitions: 8](#_Toc167006726)

[Invariants: 8](#_Toc167006727)

[TICKET TIMED AUTOMATA 8](#_Toc167006728)

[Locations: 9](#_Toc167006729)

[Clocks: 9](#_Toc167006730)

[Channels: 9](#_Toc167006731)

[Transitions: 10](#_Toc167006732)

[Invariants: 11](#_Toc167006733)

[VEHICLE TIMED AUTOMATA 11](#_Toc167006734)

[Local Declarations 11](#_Toc167006735)

[Locations: 12](#_Toc167006736)

[Clocks: 12](#_Toc167006737)

[Channels: 12](#_Toc167006738)

[Transitions: 12](#_Toc167006739)

[PAYMENT TIMED AUTOMATA 14](#_Toc167006740)

[Locations: 14](#_Toc167006741)

[Clocks: 14](#_Toc167006742)

[Channels: 14](#_Toc167006743)

[Transitions: 14](#_Toc167006744)

[Invariants: 16](#_Toc167006745)

[Declarations 16](#_Toc167006746)

[System Declarations 16](#_Toc167006747)

[UPPAAL Simulation 17](#_Toc167006748)

[Swimlanes 26](#_Toc167006749)

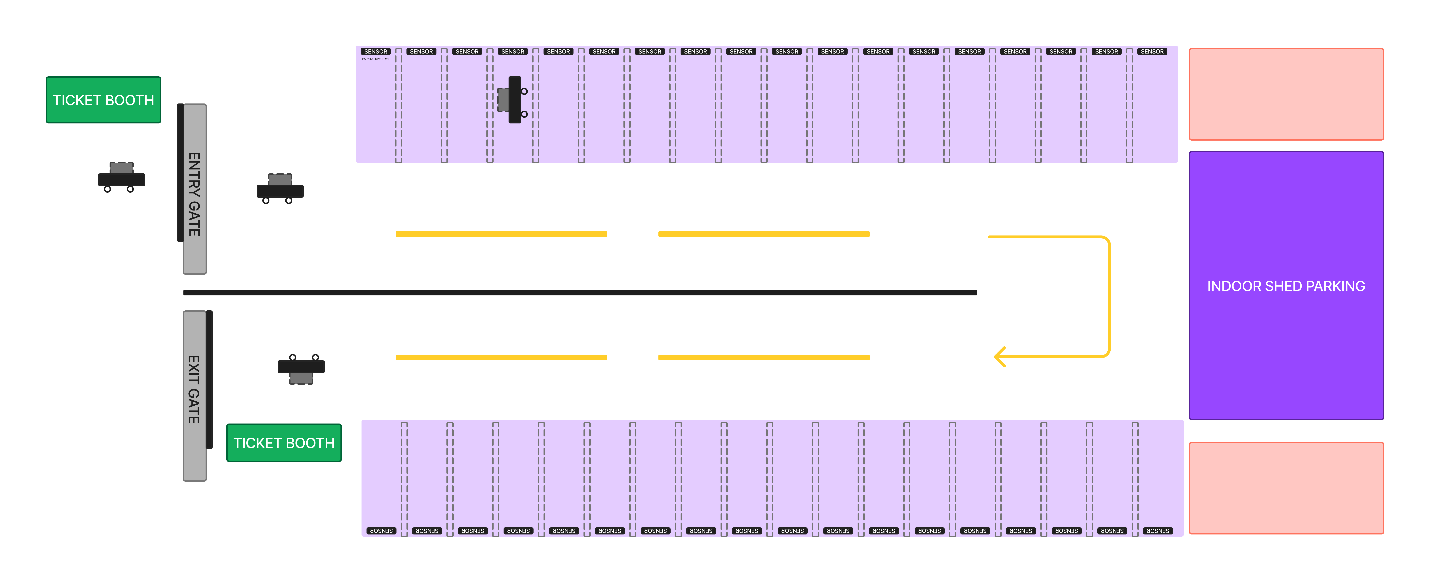
[CONCLUSION 28](#_Toc167006750)

# Introduction:

In the era of rapid urbanization and technological advancement, the demand for efficient parking systems has surged. To address this need, smart car parking systems have emerged as a promising solution, offering enhanced convenience, safety, and sustainability. This document presents an in-depth exploration of timed automata modeling applied to various components of a smart car parking system deployed at the National University of Sciences and Technology (NUST). Timed automata, a formalism rooted in theoretical computer science, provide a powerful framework for modeling and analyzing the dynamic behavior of complex systems with temporal constraints. By employing timed automata, we aim to capture the intricacies of the smart parking system, including vehicle entry and exit management, parking space allocation, and real-time monitoring.

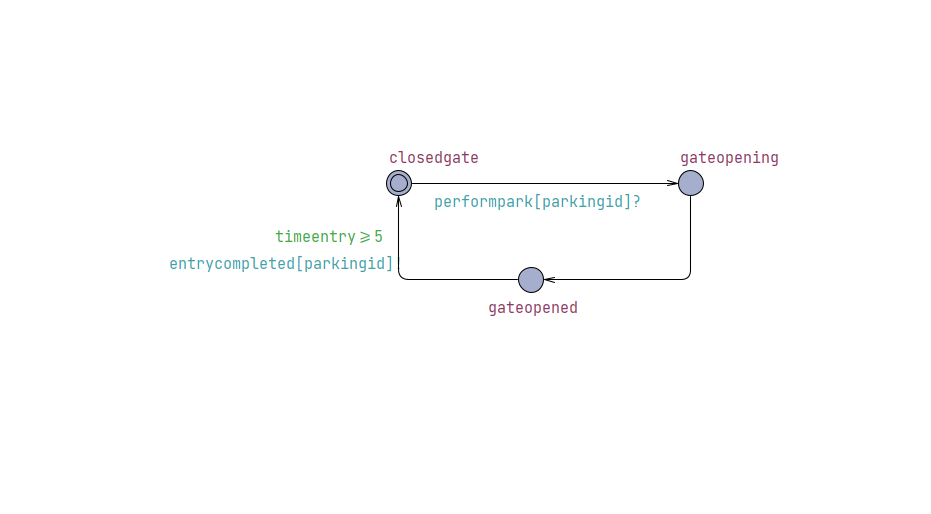
# System Overview

The NUST Smart Parking System revolutionizes the parking experience through an integrated approach to management and automation. As vehicles approach the entry gate, they are seamlessly guided through the ticketing process, with options for payment via cash or a parking card. Crucially, the system continuously monitors parking space availability using sensors deployed in each space, ensuring that users are directed to vacant spots upon entry. Once payment is complete, vehicles are granted access, and entry gates open automatically, allowing smooth entry into the facility. Through sophisticated algorithms, the system allocates parking spaces efficiently, minimizing congestion and maximizing space utilization. Each parking ticket comes with a predefined duration, typically 60 minutes, after which vehicles must exit the parking lot. Failure to do so results in automatic blocking of the parking space until an overtime fine is paid by the driver. This fine must be settled before the vehicle can exit the facility through the exit gate. Centralized monitoring and management systems oversee the entire operation, collecting real-time data on space availability, occupancy, and transactions, enabling proactive management and rapid resolution of any issues. Overall, the NUST Smart Parking System offers a seamless, user-friendly experience while optimizing space usage and enhancing operational efficiency.



# Timed Automata Modeling

## PARKING LOT ENTRY GATE TIMED AUTOMATA



### Locations:

1. Gate Closed – Parking Lot Entry Gate (initial location)
2. Gate Opening
3. Gate Open

### Clocks:

Timeentry (continuous clock representing the passage of time)

### Channels:

* Performpark[parkingid]? : It is a signal received by the system which basically tells it to open the parking gate.
* Entrycompleted[parkingid]! : It is released by the parking gate to tell that the car has successfully parked.

### Transitions:

1. **Transition from Gate Closed to Opening Gate :**

* Synchronization: performpark[parkingid]?
* Reset: Time == 0

1. **Transition from Opening Gate to Open Gate:**

* Reset: Time ==0

1. **Transition from Open Gate to Gate Closed:**

* Synchronization: Vehicle Entry Complete
* Guard: Time >= 5 sec

### Invariants:

1. Gate Closed: None
2. Gate Open: None
3. Opening Gate: time < 5

## PARKING LOT EXIT GATE TIMED AUTOMATA

**A diagram of a network

Description automatically generated**

### Locations:

1. Gate Closed – Parking Lot Exit Gate
2. Gate Opening
3. Gate Open

### Clocks:

Time (continuous clock representing the passage of time)

### Channels:

* Performunpark[parkingid]? : It is a signal received by the system which basically tells to open the parking gate.
* Exitcompleted[parkingid]! : it is released by the parking gate to tell that the car has left the system

### Transitions:

1. **Transition from Gate Closed to Opening Gate:**

* Synchronization: performunpark[parkingid]?
* occupiedSpace is decremented as an car leaves the Parking Lot

1. **Transition from Opening Gate to Gate Open:**

* Trigger: None
* Reset: Time == 0

1. **Transition from Open Gate to Gate Closed:**

* Synchronization: ExitCompleted signal issued
* Guard: Time >= 5 sec

### Invariants:

1. Gate Closed: None
2. Gate Open: None
3. Opening Gate: time < 5 sec

## CAR PARKING SPACE FINE TIMED AUTOMATA

***A screenshot of a computer

Description automatically generated***

### Locations:

* nofine
* timeexceeded

### Clocks:

None

### Channels:

* Alert[parkingid]? : It is an alert generated by the system that the valid parking time (60 minutes) is overdue.
* Finepaid[parkingid]! : it is a signal emitted that the fine has been paid. It is received by the car automata and the car can thus exit the system.

### Transitions:

**1.Transition from nofine to timeexceeded:**

* Trigger: alert[parkingid]?

**2.Transition from timeexceeded to nofine:**

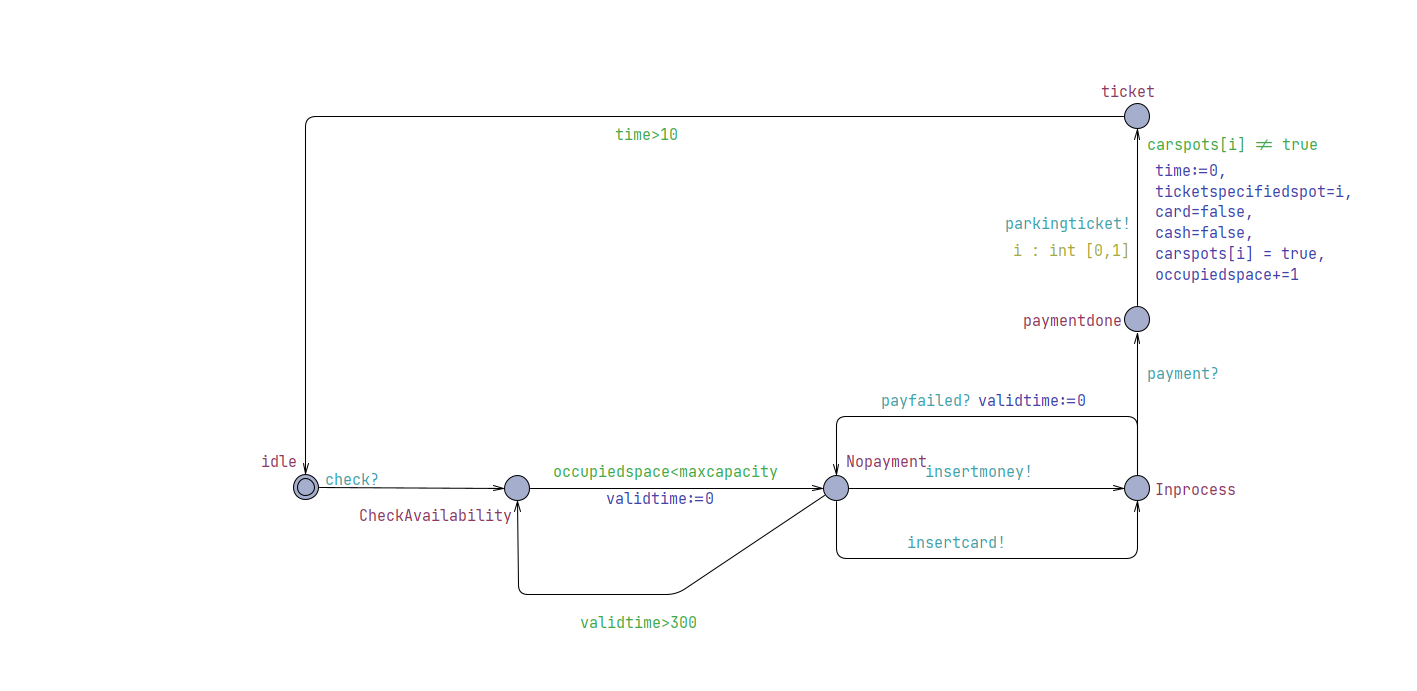
* Syncronization: finepaid[parkingid]!

So this transition is basically triggered when the fine is paid by the user.

### Invariants:

None

## TICKET TIMED AUTOMATA



### Locations:

* idle
* CheckAvailabilty
* NoPayment
* InProcess
* paymentdone
* Ticket

### Clocks:

* Time (continuous clock representing the passage of time)
* Validation time (continuous clock representing max limit for payment processing)

### Channels:

* Insertmoney! : tells that user inserted money
* Insertcard! : tells that user inserted cash
* Parkingticket! : issues the aprking ticket to the user
* payment? : receives the signal that payment has been done
* Leftparking? : receives the signal that the user has left parking and thus could enter again.

### Transitions:

1. **Transition from idle to CheckAvailability**

* Trigger: Signal from a car indicating a request for a parking space (check?).
* Action: Move to the CheckAvailability state to verify if there are available parking spots.

1. **Transition from Check Availability to No Payment:**

* Trigger: Vehicle Approaches and Input signal to sensor
* Guard: OccupiedSpace < MaxCapacity
* Reset: ValidationTime == 0

1. **Transition from NoPayment to Check Availability:**

* Trigger: ValidationTime
* Guard:ValidationTime > 300

1. **Transitions from NoPayment to InProcess :**

* Triggers: InsertMoney OR InsertCard
* Reset: Time = 0

1. **Transitions from InProcess to NoPayment :**
   * Trigger: payfailed?
   * Reset: ValidTime = 0
2. **Transition from InProcess to payment done :**

* Trigger: payment? channel.
* Reset: time = 0
* Action: Upon successful payment, move to paymentdone state.

1. **Transition from payment done to ticket:**

* Trigger: None
* Synchronization: parkingticket!
* Select: Ticket number from the range [0,1] (i.e., two parking spots).
* Update:
  + time = 0
  + Reset cash and card flags.
  + Assign the parking spot to the car.
  + Increment the occupiedspace counter.
* Action: Issue the parking ticket and move to the ticket state.

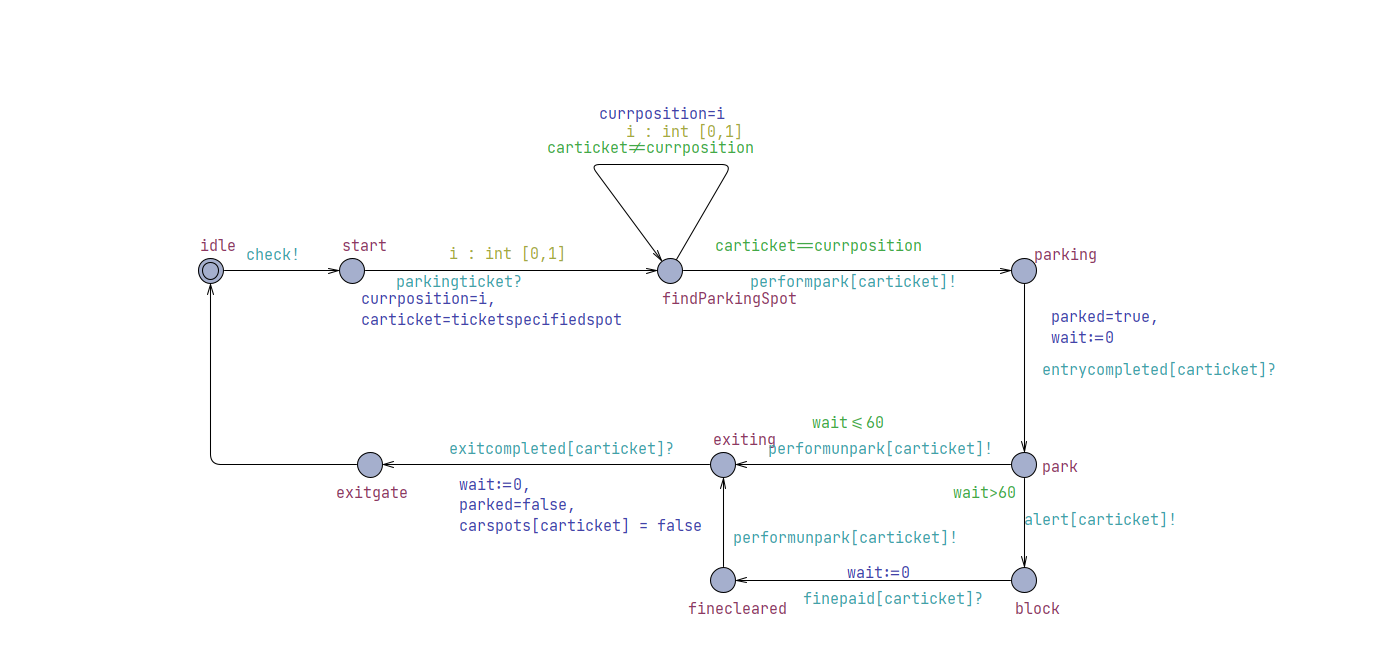
1. **Transition from ticket to idle :**

* Trigger: parkingticket! channel.
* Reset: time = 0
* Action: After issuing the ticket, return to the idle state.

### Invariants:

InProcess : Time < 5 sec

## VEHICLE TIMED AUTOMATA



### Local Declarations

* **carticket**: Represents the ticket assigned to the car.
* **currposition**: Current position of the car in the parking process.
* **wait**: A clock variable to measure waiting time.
* **parked**: A boolean to indicate if the car is currently parked.

### Locations:

* idle
* Start
* FindParkingSpot
* parking
* Park
* exiting
* Block
* finecleared
* Exit Gate

### Clocks:

* Wait (continuous clock representing waiting time)

### Channels:

* Parkingticket?: signal which tells if the user has received a parking ticket
* Performpark!: signal which tells if the user can park
* Entrycompleted?: signal which tells if entry into the parking has been completed by the user
* Performunpark!: signal which tells the system that the user wants to leave their space.
* Exitcompleted?: signal which tells if exit from the parking space has been completed by the user
* Leftparking! : signal issued when user leaves the parking
* Alert!: signal issued to tell that the user has overstayed their allocated time.
* Finepaid!: signal issued to tell that the user has paid their fine.

### Transitions:

1. **Idle to start:**

* From idle to start using the check! synchronization channel.
* Syncronization: check!

1. **Transition from Start to FindParkingSpot:**

* Trigger: parkingTicket?
* Select: parking space number from 1-2 at which vehicle currently is.

1. **Transition from FindParkingSpot to FindParkingSpot:**

* Guard: carticket != currposition
* Select: number of parking space adjacent to the current one.

1. **Transition from FindParkingSpot to Parking:**

* Trigger: performpark[carticket]!
* Guard: carticket == currposition

1. **Transition from Parking to Park:**

* Trigger: entrycompleted[carticket]?
* Guard:Parked = true
* Reset: wait = 0

1. **Transition from Park to Block:**

* Trigger: wait time is more than 60
* Guard: wait > 60 min
* Synchronization: alert[carticket]!

1. **Transition from Park to Exiting:**

* Trigger: performunpark[carticket]!
* Guard: wait <= 60 min

1. **Transition from Exiting to Exit\_Gate:**

* Trigger: exitcompleted[carticket]?
* Guard: parked = false
* Reset: wait == 0

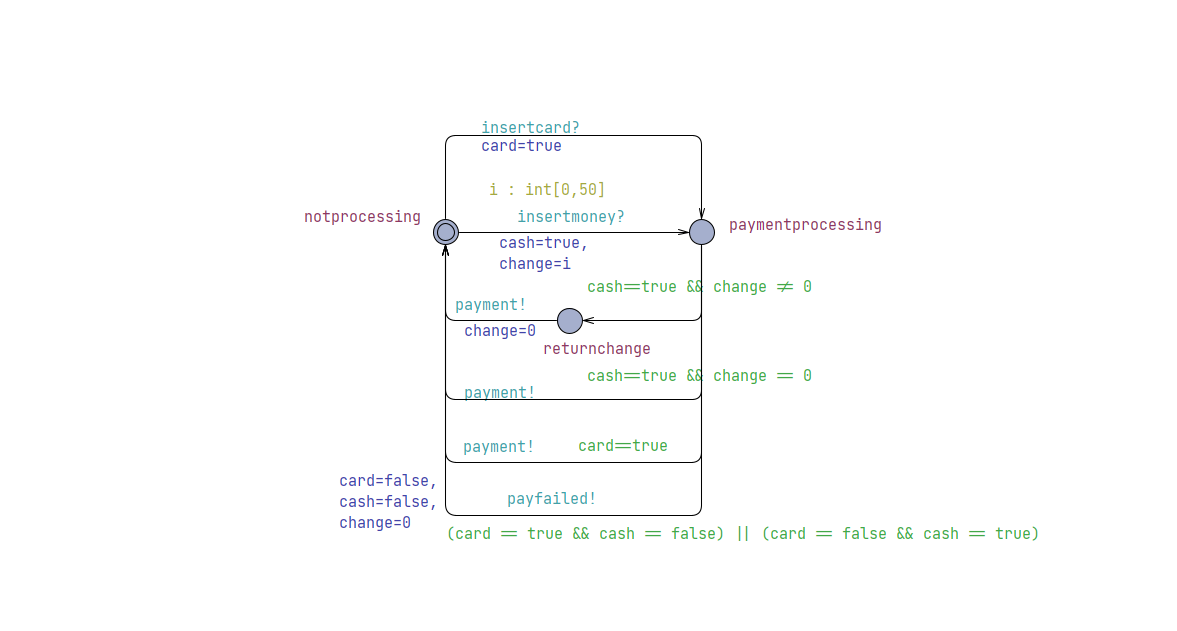
1. **Transition from Block to finecleared:**

* Trigger: Finepaid[carticket] signal from fine automata
* update: wait=0

1. **Transition from finecleared to Exiting :**

* Trigger: performunpark[carticket]!

## PAYMENT TIMED AUTOMATA

******

### Locations:

* notprocessing
* Paymentprocessing
* nochange

### Clocks:

None

### Channels:

* insertcard? : It is a signal if the user gives a card for payment.
* Insertcash?: It is a signal if the user gives cash for payment.
* Payment!: signal issued when payment is successful
* payfailed! : it is a signal emitted that the payment has failed.

### Transitions:

**1.Transition from noprocessing to paymentprocessing:**

* Triggers: insertmoney?
* Select: value of Change from 1-50
* Update: cash==true

**2.Transition from noprocessing to paymentprocessing:**

* Triggers: insertcard?
* Update: cash==true

**3.Transition from paymentprocessing to noprocessing :**

* Syncronization: payment!
* Guard: cash==true and change ==0

So this transition is basically triggered when cash payment is done but there is no change to be returned.

**4.Transition from paymentprocessing to noprocessing :**

* Syncronization: payment!
* Guard: card==true

So this transition is basically triggered when card payment is done

**5.Transition from paymentprocessing to nochange:**

* Trigger: change!=0
* Guard: cash==true and change !=0

So this transition is basically triggered when cash payment is done and there is change to be returned.

**6.Transition from nochange to noprocessing :**

* Syncronization : payment!
* Update : change = 0

So this transition is basically triggered when change is returned and ticket is issued

**7. Transition from paymentprocessing to noprocessing :**

* Trigger: failure of payment for some reason
* Guards: card == true XOR cash == true
* Reset: Change= 0, card = false and cash = false
* Effect: payfailed! Signal issued

### Invariants:

None

## Declarations

A computer screen shot of a computer code

Description automatically generated

Car local declarations

A screenshot of a computer

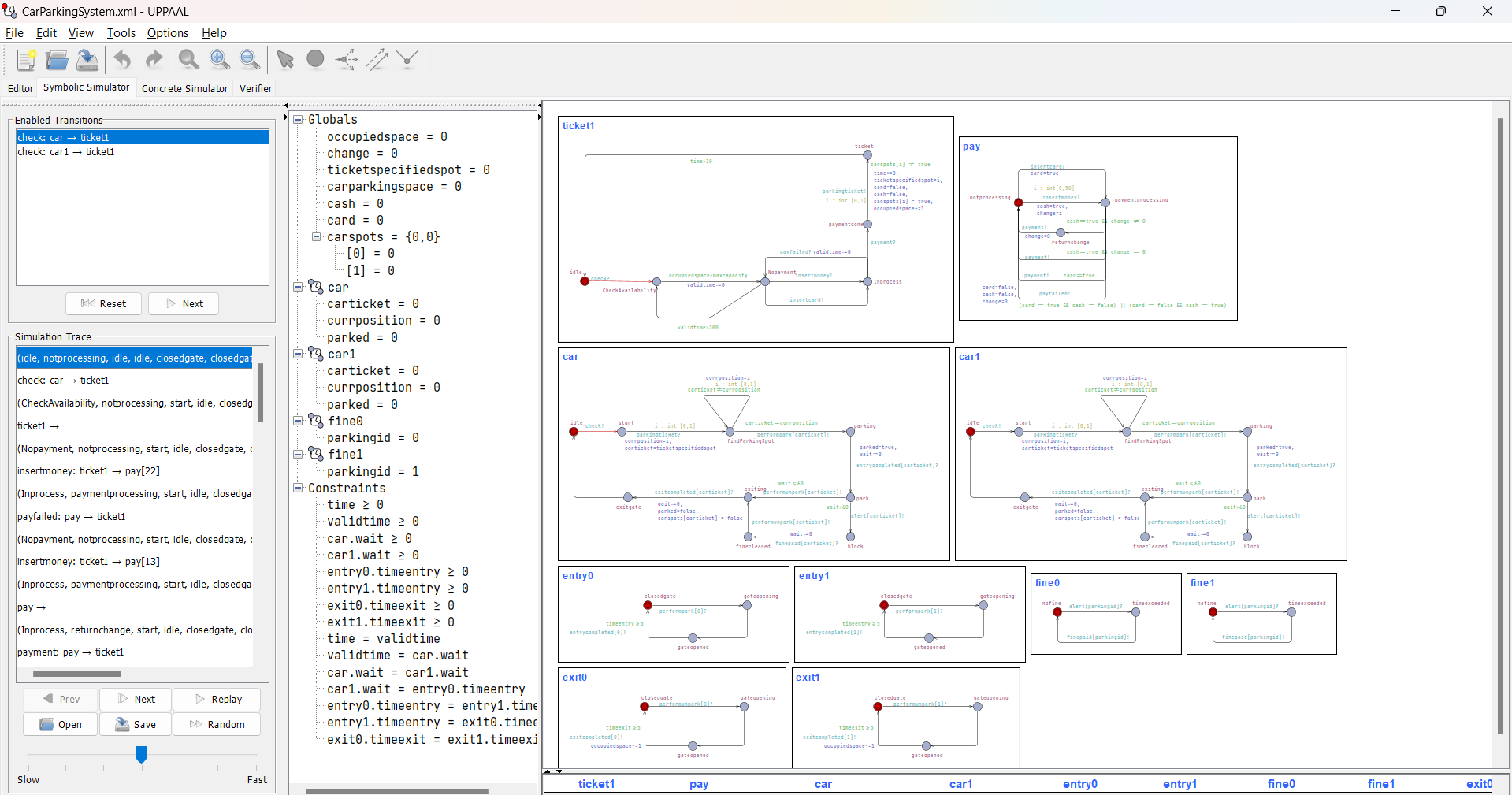
Description automatically generated

## System Declarations

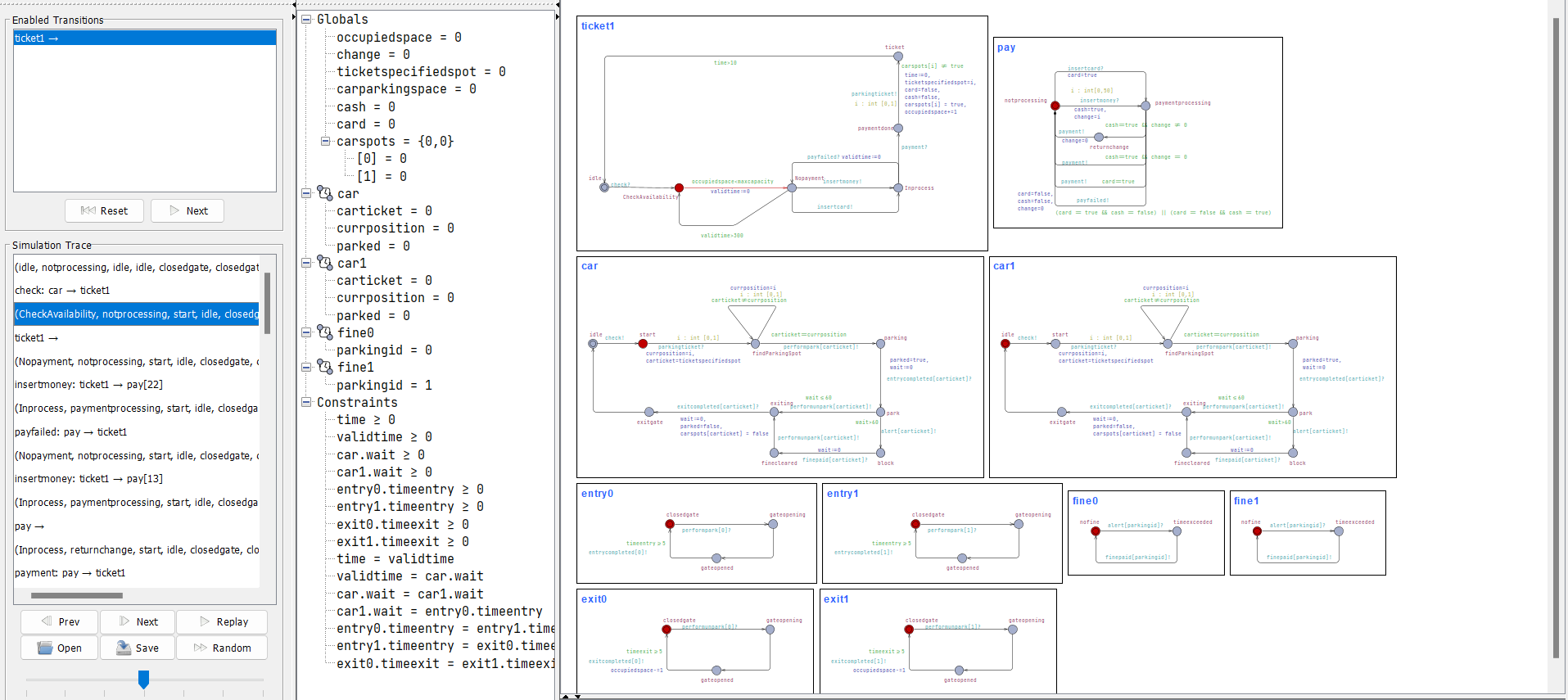
A screenshot of a computer

Description automatically generated

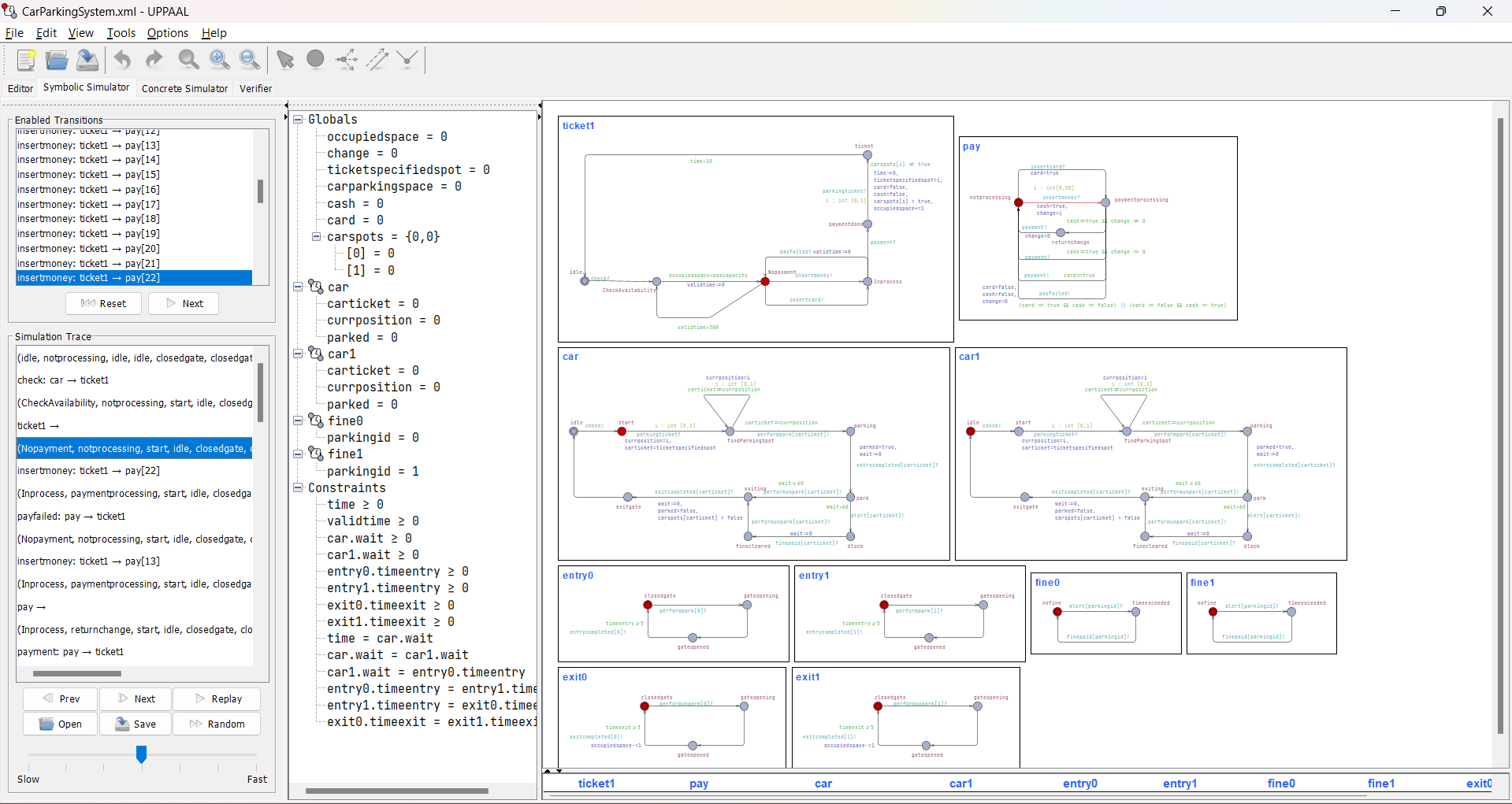
## UPPAAL Simulation



Car1 is sending signal to check for availibility.



Car1 is at the gate and checking for the availability of space



Space is available and the user moves to nopayment

A screenshot of a computer

Description automatically generated

User pays through cash and has rs 22 change to be returned.

A screenshot of a computer

Description automatically generated

Payment is failed and the user is going to retry with a different note, now change to be returned is RS 13.

A screenshot of a computer

Description automatically generated

User change is returned and payment is successful

A screenshot of a computer program

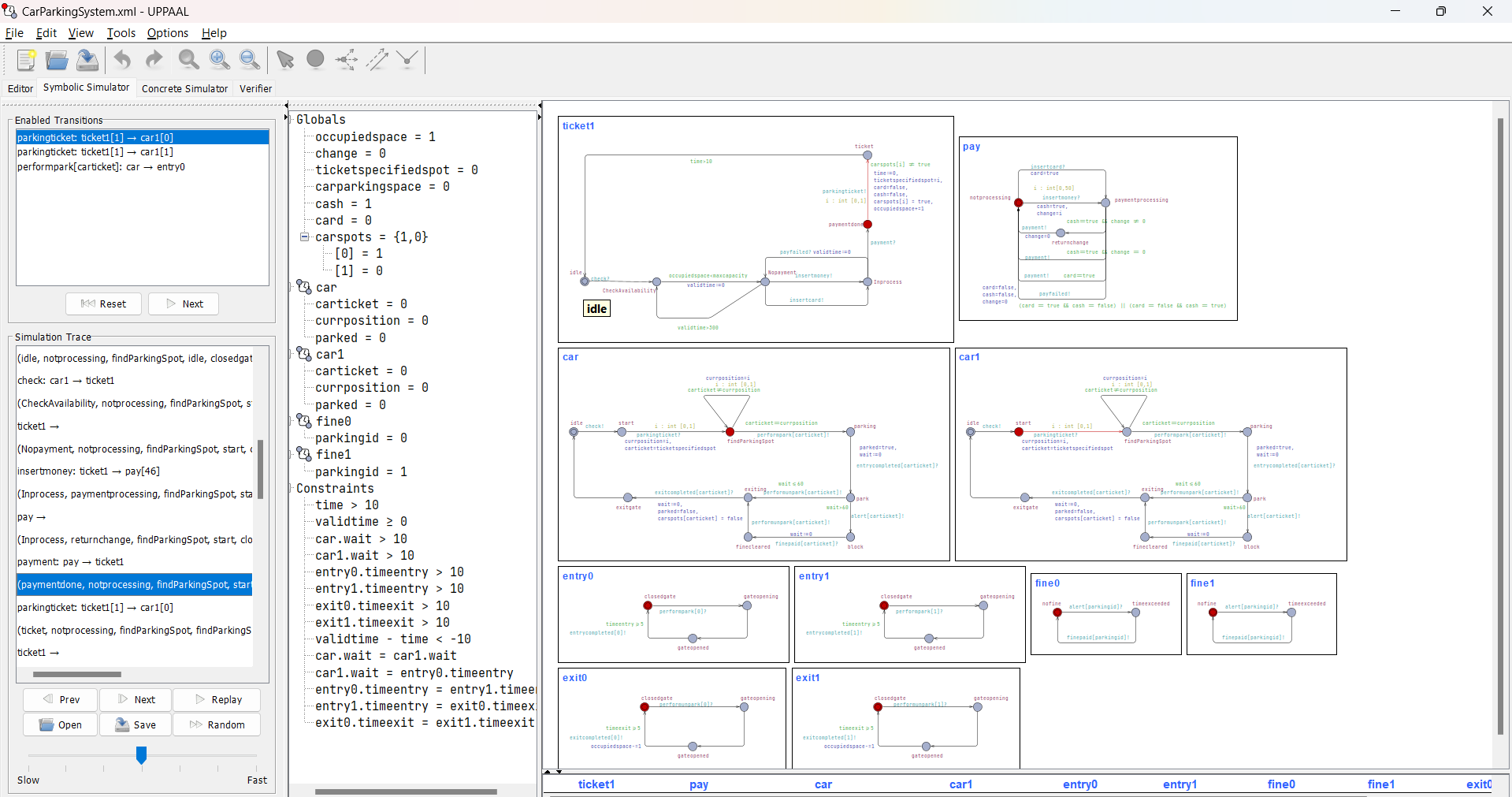
Description automatically generated

Payment module has returned to noprocessing following a successful payment and ticket module is at payment stage and generating a ticket.

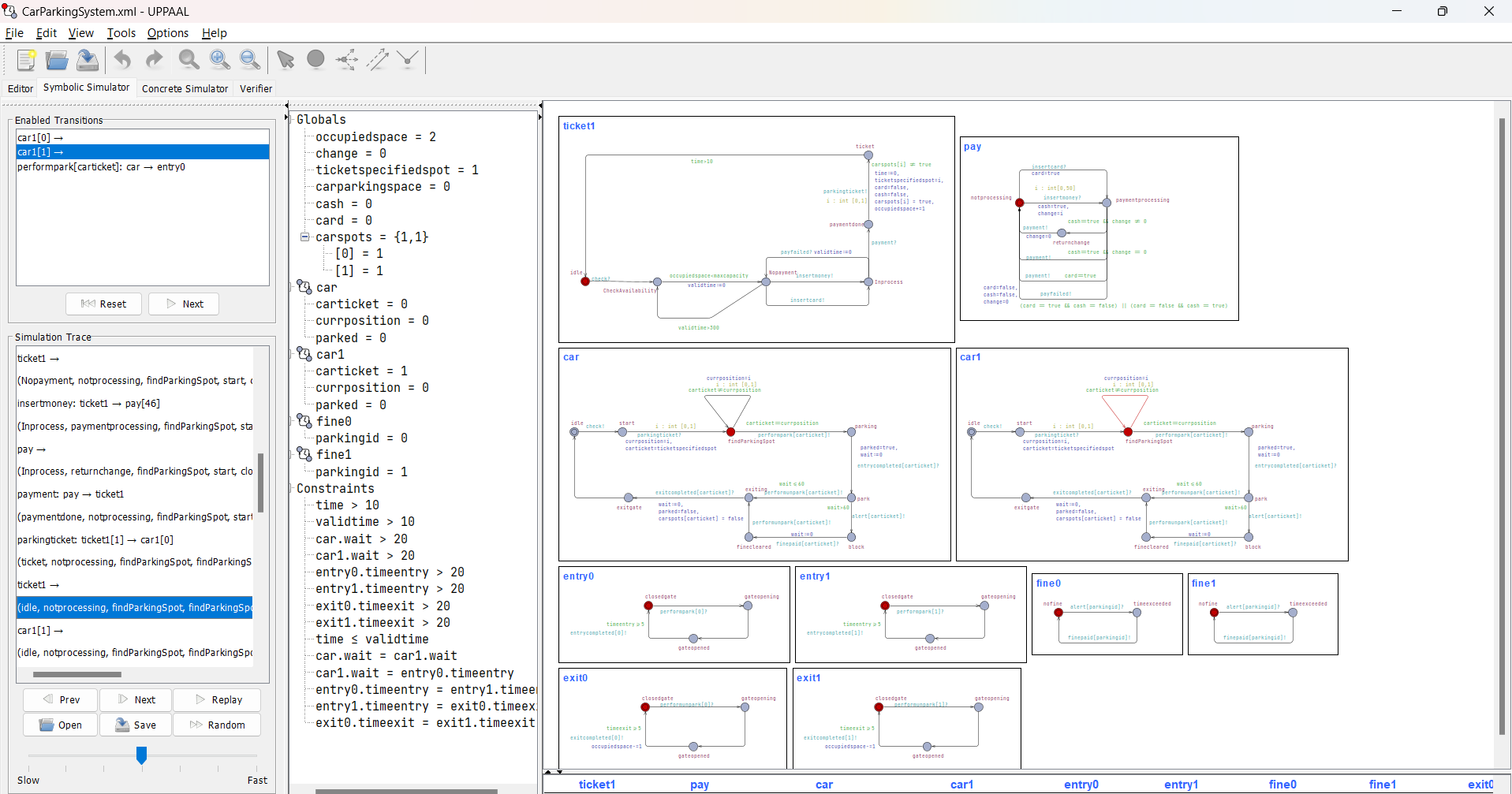
A screenshot of a computer

Description automatically generated

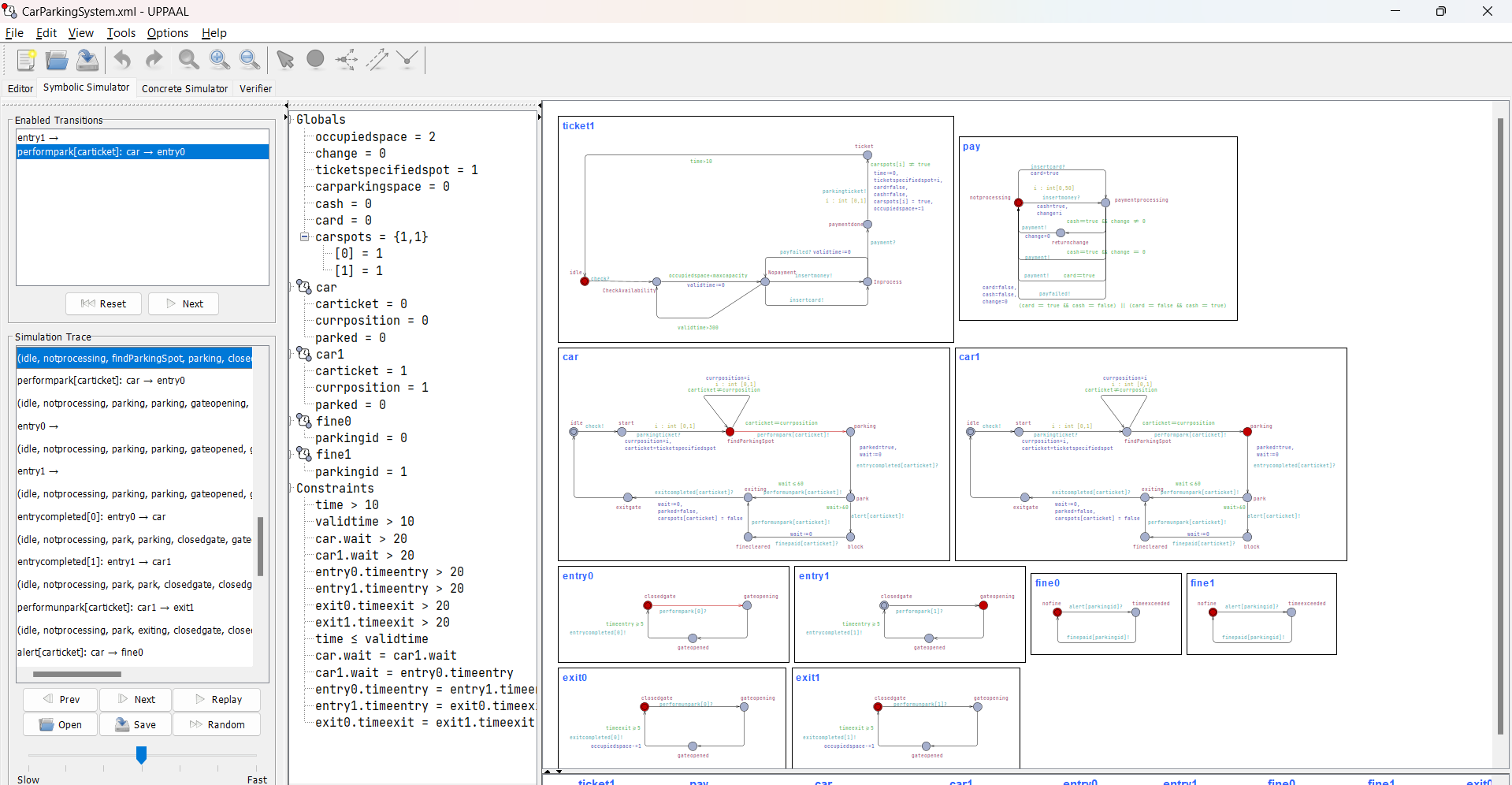
Car got ticket id 0 and is at the space, meanwhile another car is at the teicketing booth checking for space.



Second car pays for the ticket and gets a ticket of 1 as only that spot is available. It is currently at spot 0.



Second car tries to find the spot whereas first car is yet to start the parking process



Second car ahs found the spot whereas first car is parking

A screenshot of a computer

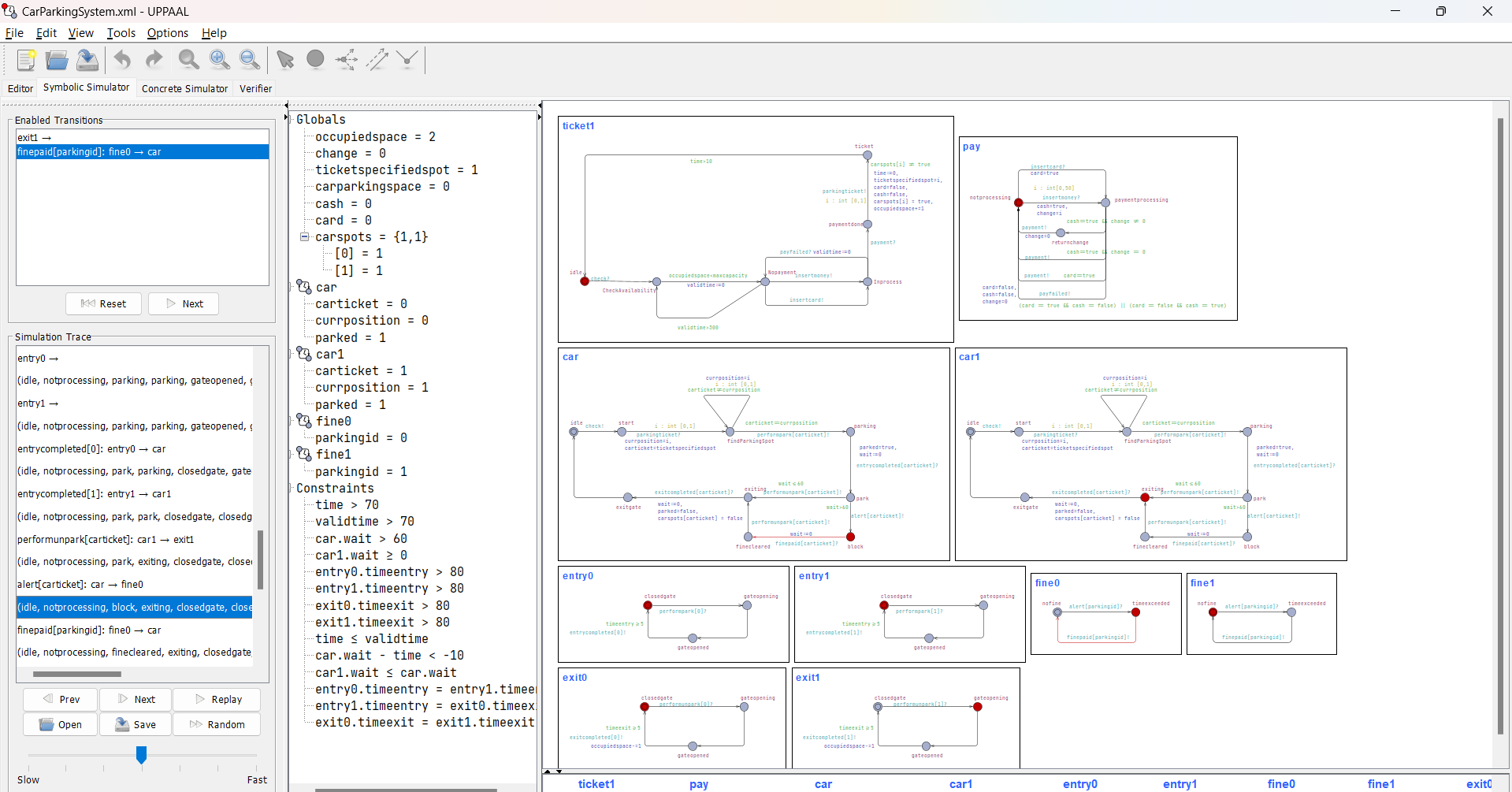
Description automatically generated

Both cars have started the parking process together and the gates for both the parking spots are opening.

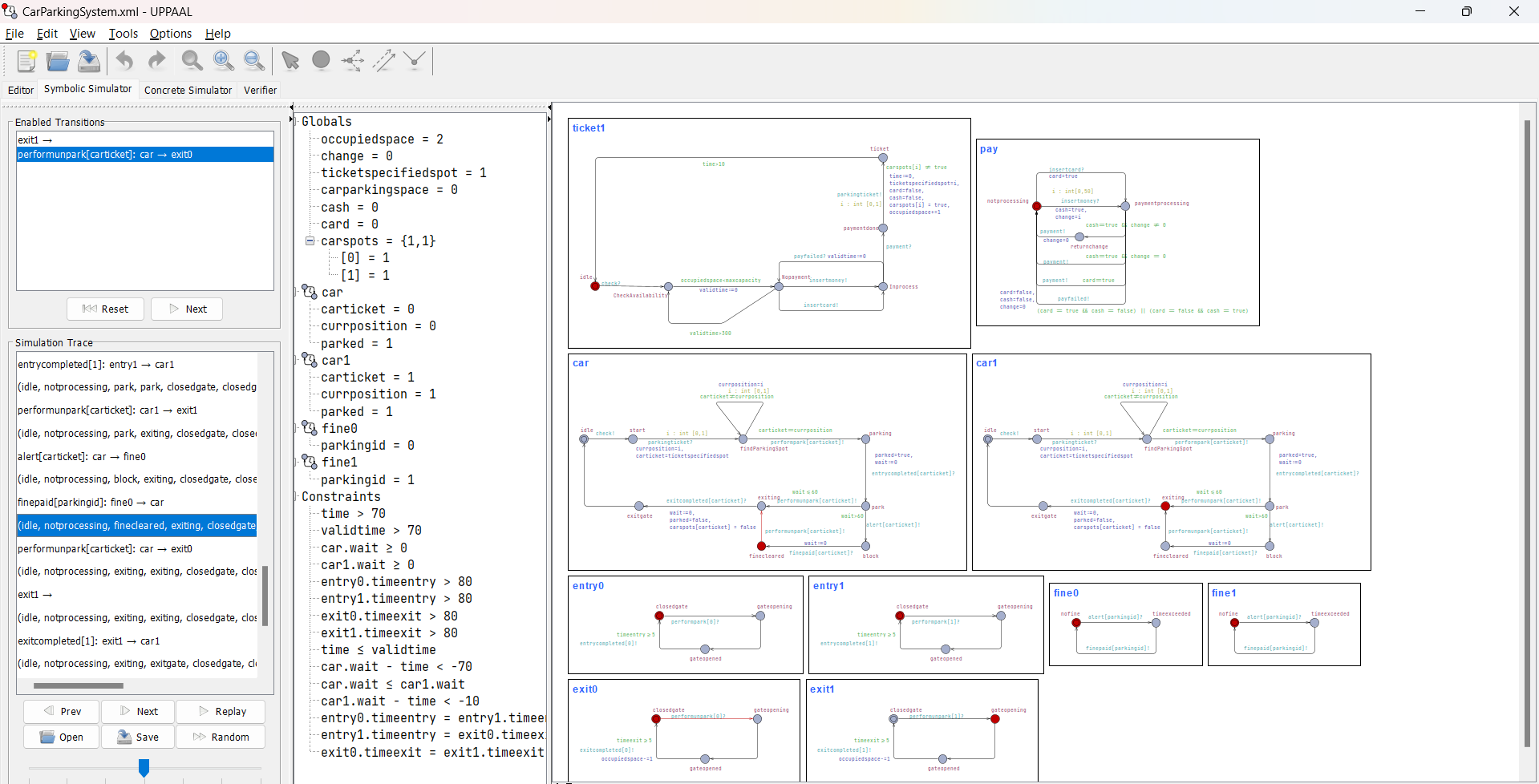
A screenshot of a computer

Description automatically generated

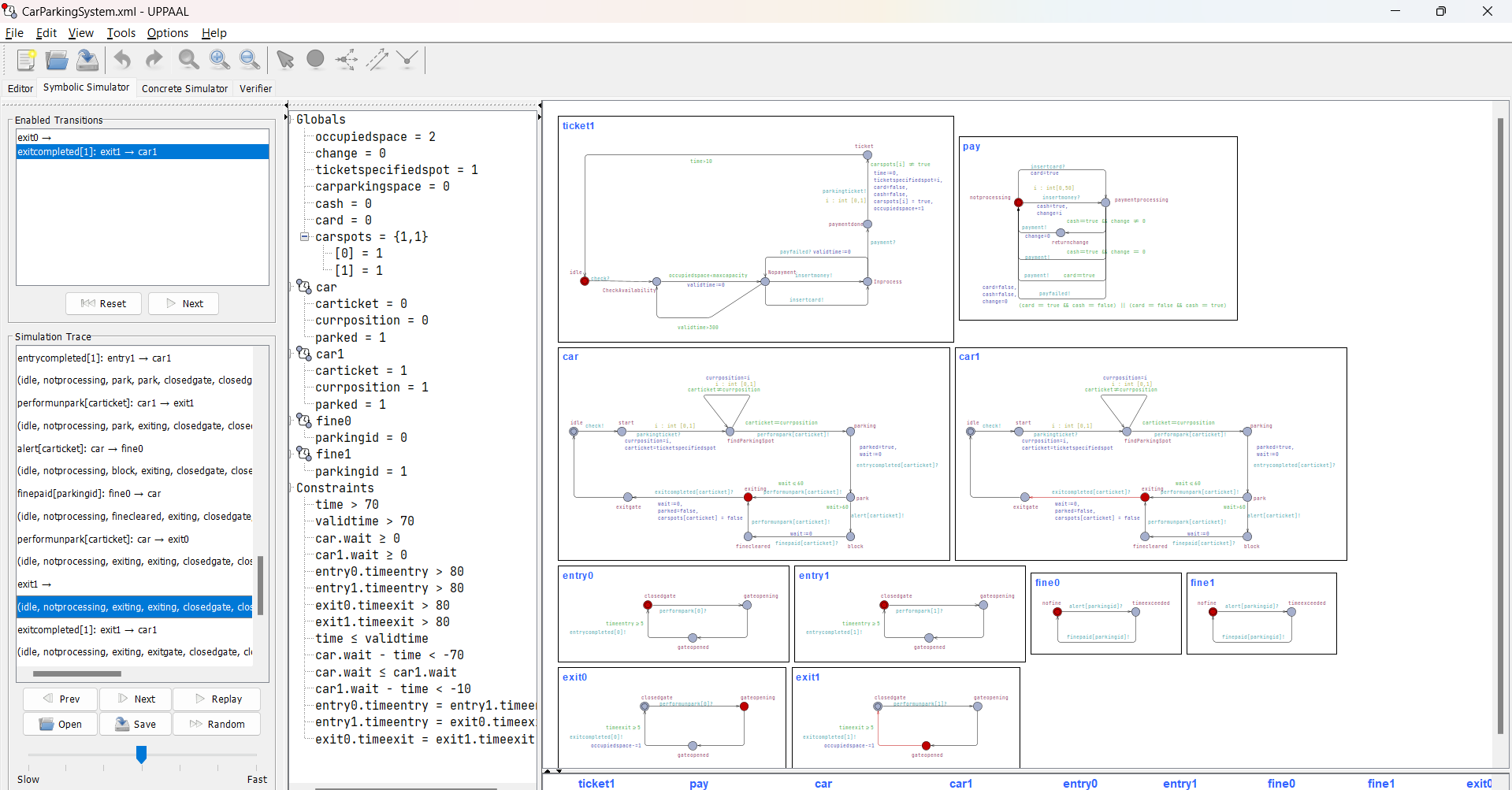
Both cars have parked successfully



Car1 has initiated the exit process and the exit gate for that spot is openeing whereas car has overstayed and is now in block state, has to pay fine



Fine has been paid whereas the other car is exiting still



Car1 parking gate has opened, wheras car has also given the signal to exit, so its gate is also openeing

A screenshot of a computer

Description automatically generated

Car1 has exited and is now in idle state whereas car is exiting still.

A screenshot of a computer

Description automatically generated

Both cars in original state.

## Swimlanes

A screenshot of a computer program

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

# CONCLUSION

In conclusion, the application of timed automata modeling to the smart car parking system at NUST has yielded valuable insights into its operation, performance, and potential for optimization. By leveraging the formalism of timed automata we have been able to capture the system's temporal dynamics with precision, enabling thorough analysis and verification of its behavior under various scenarios. Our findings underscore the effectiveness of timed automata as a modeling tool for complex real-world systems, facilitating the identification of bottlenecks, optimization opportunities, and resilience-enhancing measures.