Final Version of Project

Automatic Sliding Door System

**Group Members**

kashif KHAN – 209147

m hamza zahid – 219400

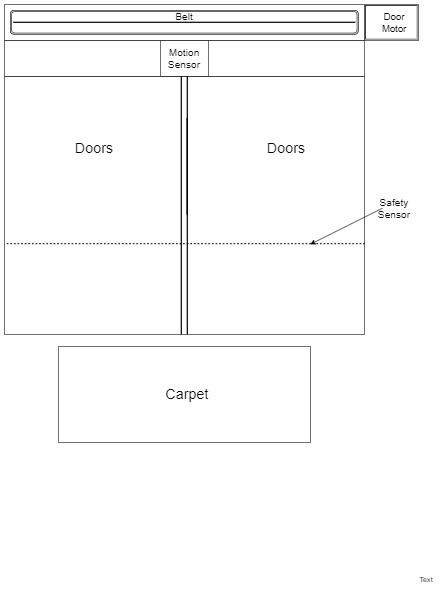
BESE-8B

2020

**1. Project Description:**

Automatic Sliding Doors are used to detect a person by sensor and open automatically instead the person opens the door manually. When a person take step on carpet a signal is send to door which than send signal to motor to start and doors start to open. If person is standing between the doors, the safety sensor keeps the doors open.

**Block Diagram:**



**2. System Requirements:**

1. The motion sensor will be either on or off.

**µ m:Motion Sensor.SensorOn(m)⊕SensorOff(m)**

1. The safety sensor will be either on or off.

**µ s:safety Sensor.SensorOn(s)⊕SensorOff(s)**

1. The doors will open for at most 2 sec for each person.

**∀p:Person. doors\_open (p, 2)**

1. The doors will open when the person step on the carpet.

**∀p:Person|steps\_on\_carpet (p).doors\_open (p)**

1. Motion sensor will detect the person steps on the carpet

**µ m:Motion Sensor ∀p:Person|step\_on\_carpet(p).detect(m,p).**

1. The door motor will turn on if a person steps on the carpet.

**µ q:Door Motor ∀p:Person |steps\_on\_carpet(p).MotorOn (q)**

1. The door motor will run for 2 sec till the doors are opened.

**µ q:Door Motor.doors\_open()→MotorOn (q,2)**

1. The doors will re-open in case a new person steps on the carpet.

**∀n: New Person.steps\_on\_carpet(n)→doors\_open(n)**

1. The doors will close only if no person is on the carpet or if person have crossed the doors.

**∀p:Person|¬oncarpet(p)∨crossed\_the\_door(p).doors\_close()**

1. The system will only act as one way entry system from one side of door.

**∀ x:system . entryAllow(x) ⊕ exitAllow(x)**

1. Safety Sensor will keep the doors open if person is standing between the doors.

**µ s: Safety Sensor ∀p:Person|standing\_between\_the\_doors (p) → doors\_open(p)**

1. Safety sensor will not generate alarm if person is not standing between the doors.

**µ s: Safety Sensor ∀p:Person|¬between\_doors(p)→¬alarm(s)**

**As motor, motion and safety sensor are only one in the system so they are unique.**

**3.Model Checker used (UPPAL)**

We have used UPPAL model checker, which uses Java SE environment. It is simple and easy to use because we only need to construct the required automaton, give the global declarations of instances of automaton and local declarations of clock in it, and then simulate it directly. We do not need much pre requisite experience to work on it except for understanding of how to construct automaton and how to declare global variables. That is one of the reasons I chose UPPAL model checker. Secondly, we had learned many examples modelling in UPPAL such as the lift system and traffic light system so I was also in ease while using it. That is also the reason I have chosen to work with UPPAL rather than any other tool.

**4, 5.Explanation of Network Automata:**

Initially the carpet is in empty state when person step on it, carpet moves from empty state to filled state and send the ‘approaching’ signal to door upon receiving ‘approaching’ signal the door than moves from close state to order\_1 state and then from order\_1 state to opening state initializing clock ‘y’ as zero (y=0) while sending ‘on’ signal to motor upon receiving ‘on’ signal by the motor than moves from stop state to run state and also clock ‘m’ is initialize as zero (m=0). The door after 2 seconds (y<=2) moves from opening to open state on the other hand motor also moves from run to stop state after 2 seconds(m<=2) at same time. Now the door is open and motor is stopped at same time. When the person leaves the carpet, carpet moves from filled to empty state sending the ‘leaving’ signal to the door upon receiving the ‘leaving’ signal the door than moves from open state to order\_2 state and then from order\_2 state to closing state sending ‘on’ signal to motor and clock ‘y’ is set to zero (y=0) upon receiving the ‘on’ signal the motor than moves from stop state to run state and clock ‘m’ is set as zero (m=0). Now there are two cases:

**Case 1: Door moves from closing to opening state:**

If a new person step on the carpet than again door upon receiving ‘approaching’ signal send by carpet will move to opening state and then from opening state to open state and motor from run state to stop state after 2 seconds. Therefore, door will again be in open state and the motor will be in stop state now.

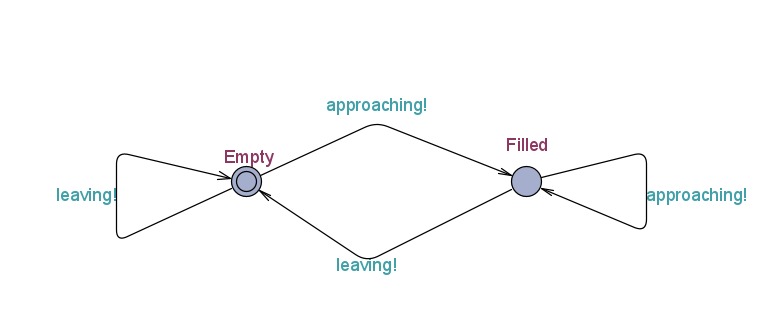
**Case 2: Door moves from closing state to close state:**

If the carpet remains empty than the door will move from closing state to close state and motor from run state to stop state after 2 seconds.

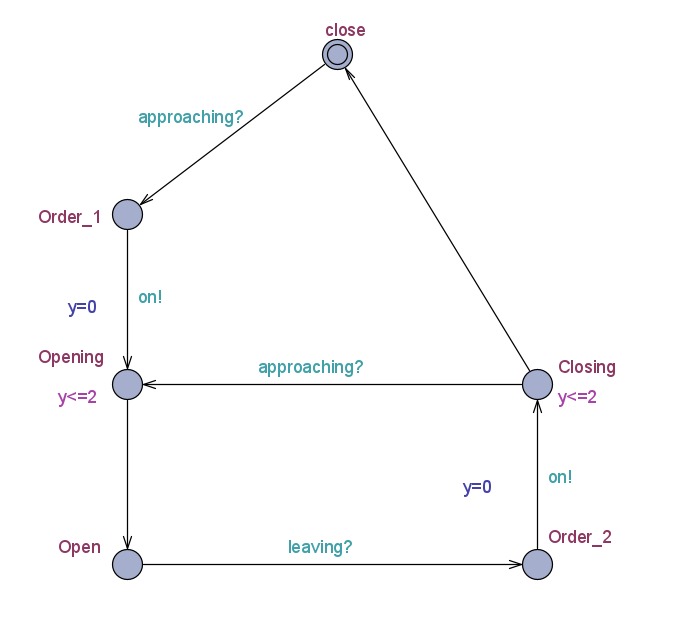
**Screen Shots:**

**Without Simulation:**

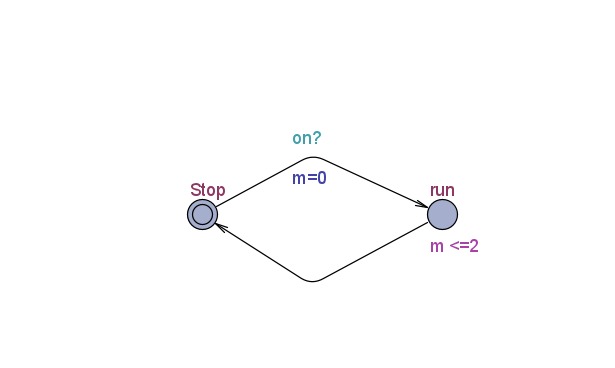
**Carpet:**



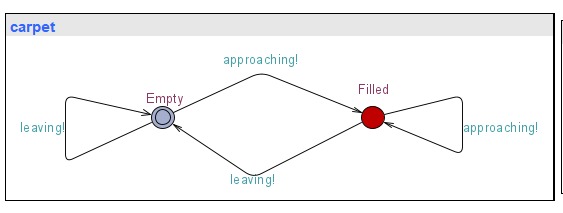
**Door:**

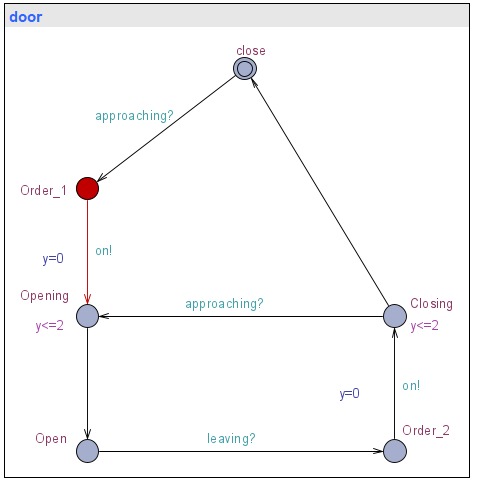


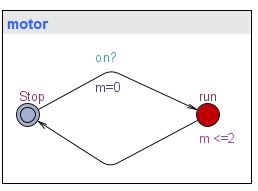
**Motor:**



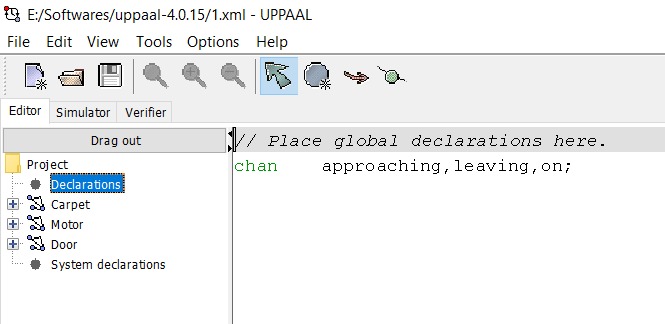
**With Simulation:**

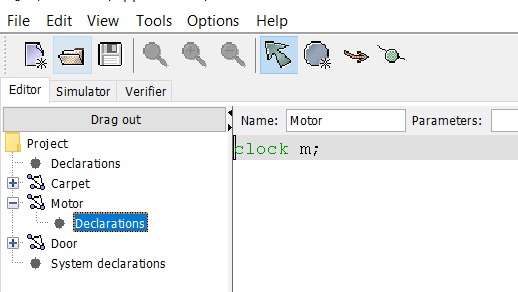


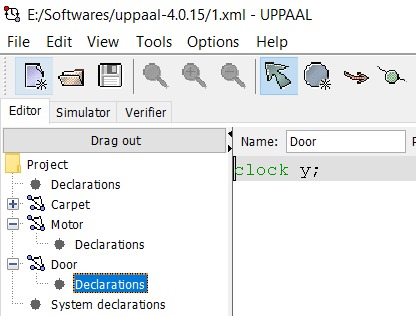


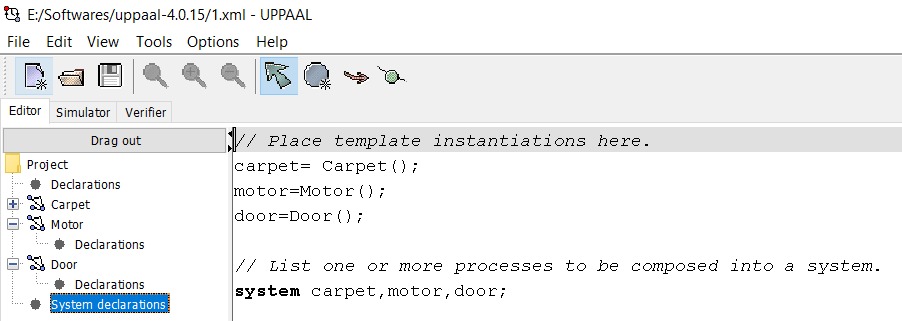


**Editor Screen Shots:**





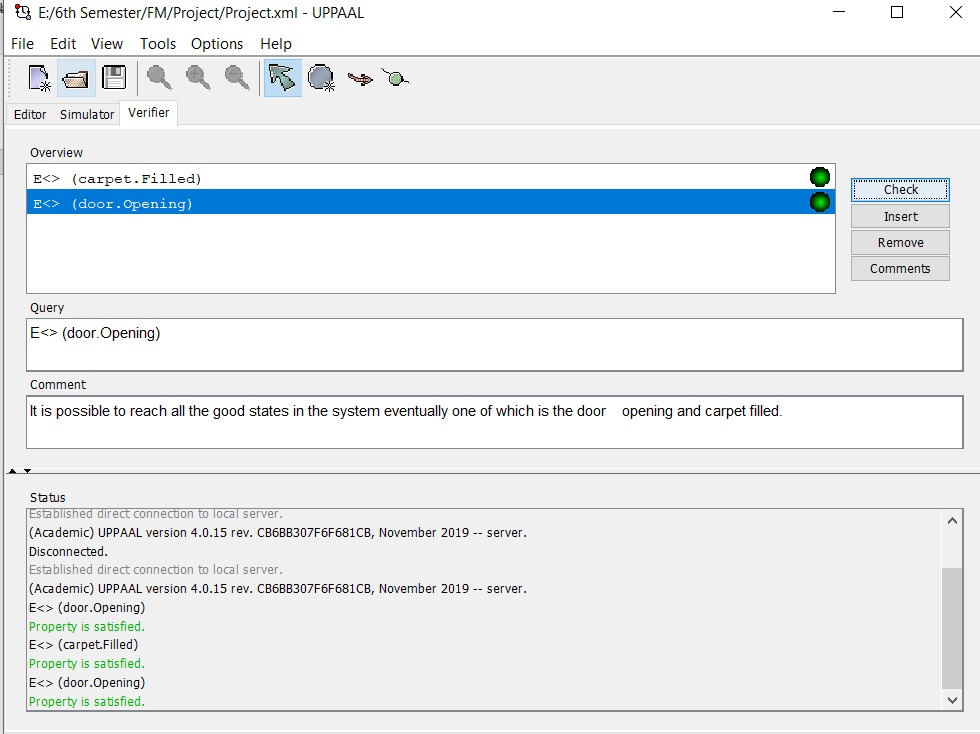




**6. Verification of system:**

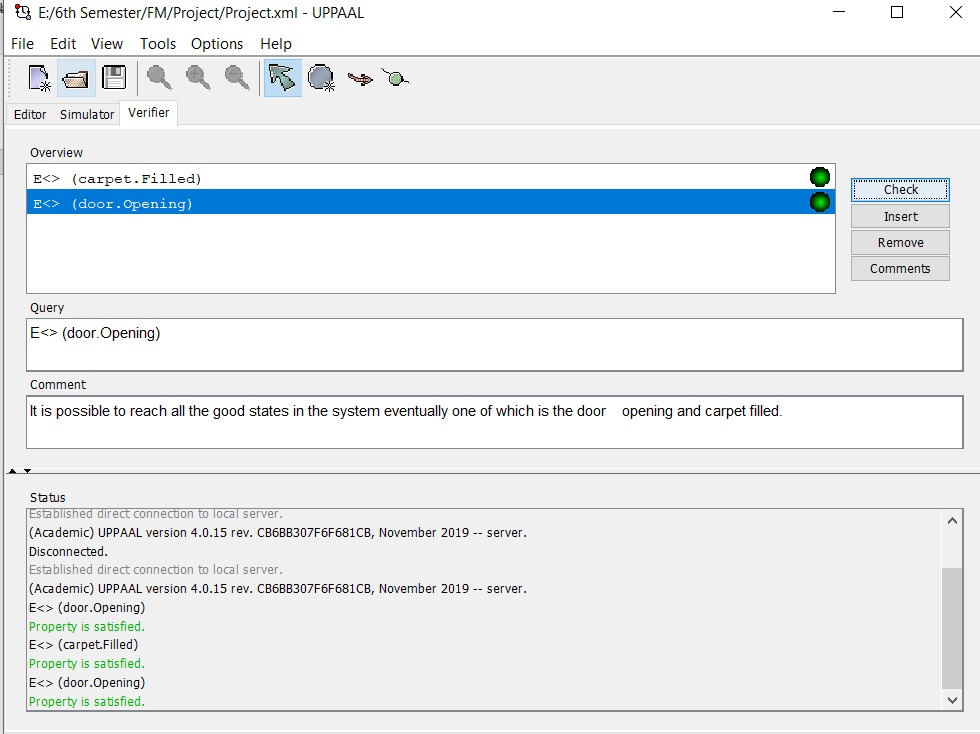
1. **Liveness:**
2. It is possible to reach all the good states in the system eventually one of which is the door opening.

**E<> (door.Opening)**

****

1. It is possible to reach all the good states in the system eventually one of which is the carpet filled.

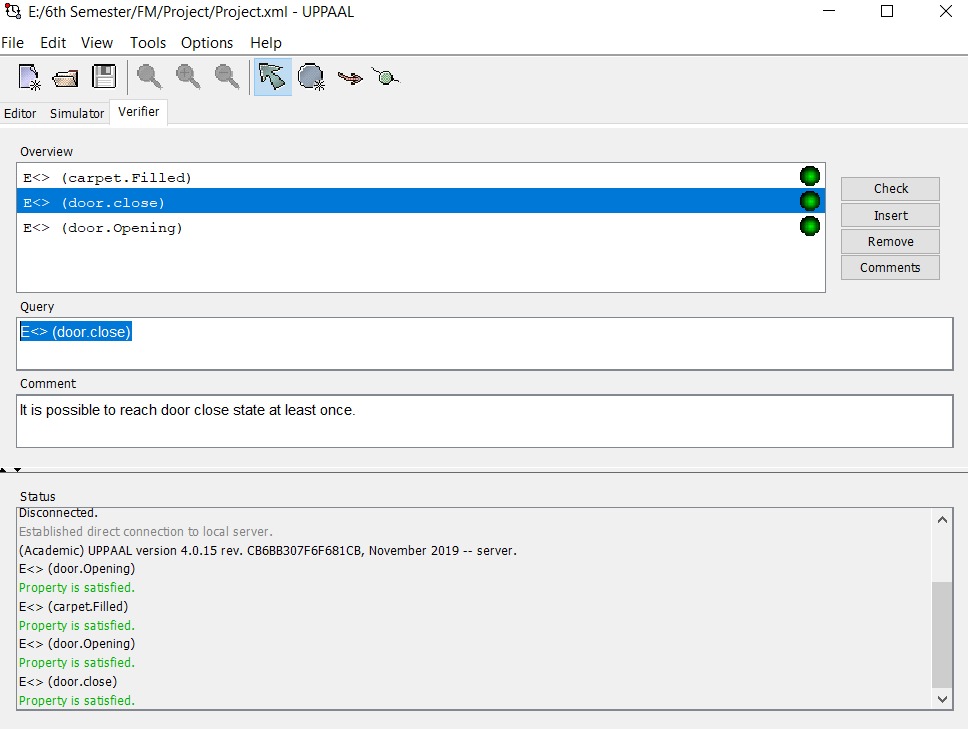
**E<> (carpet.Filled)**

****

1. **Reachability:**

It is possible to reach door close state at least once.

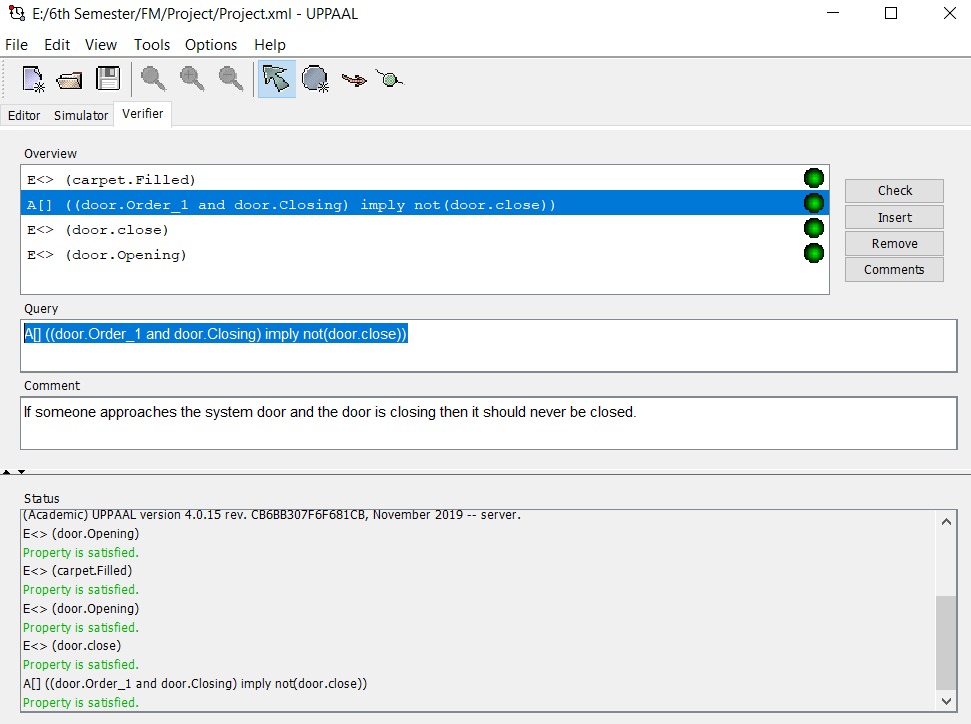
**E<> (door.close)**

****

1. **Safety:**

If person approaches the system door and the door is closing then it should never be closed.

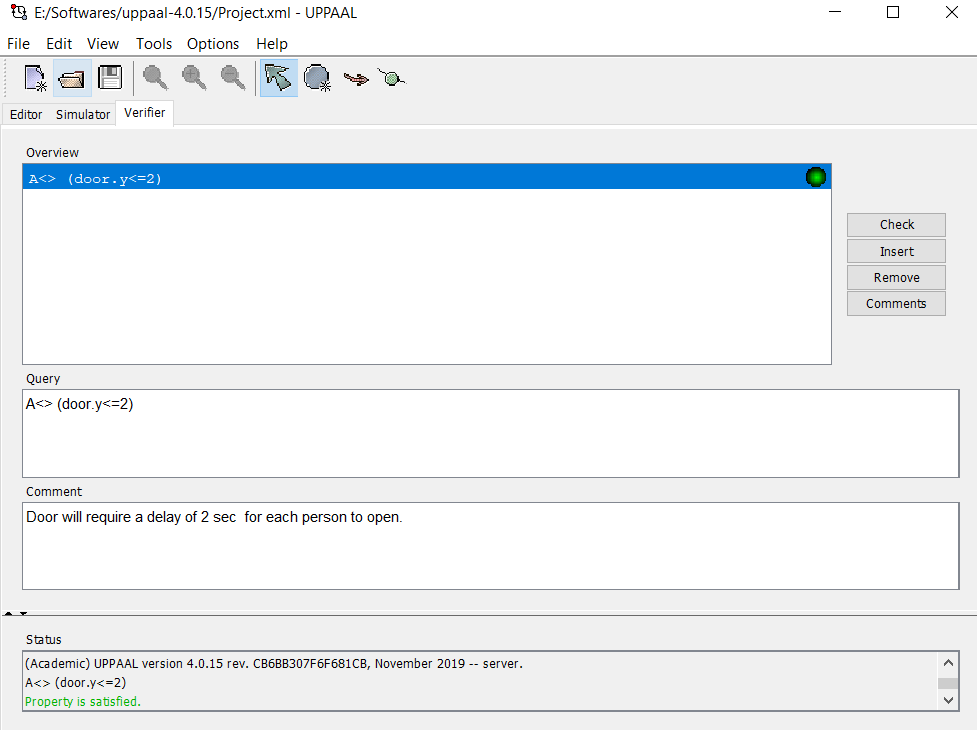
**A[] ((door.Order\_1 and door.Closing) imply not(door.close))**

****

1. **Fairness:**

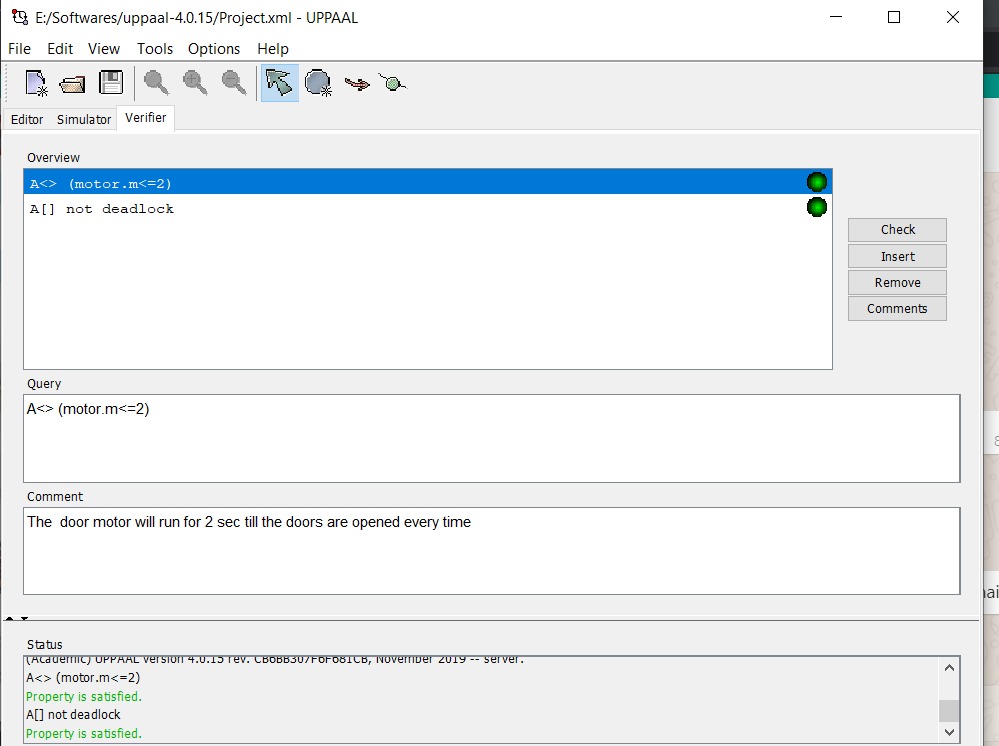
a) There should be at most a delay of 2 sec for system door to open for each person.

**A<> (door.y<=2)**



b) There should be at most a delay of 2 sec for system motor to change its state.

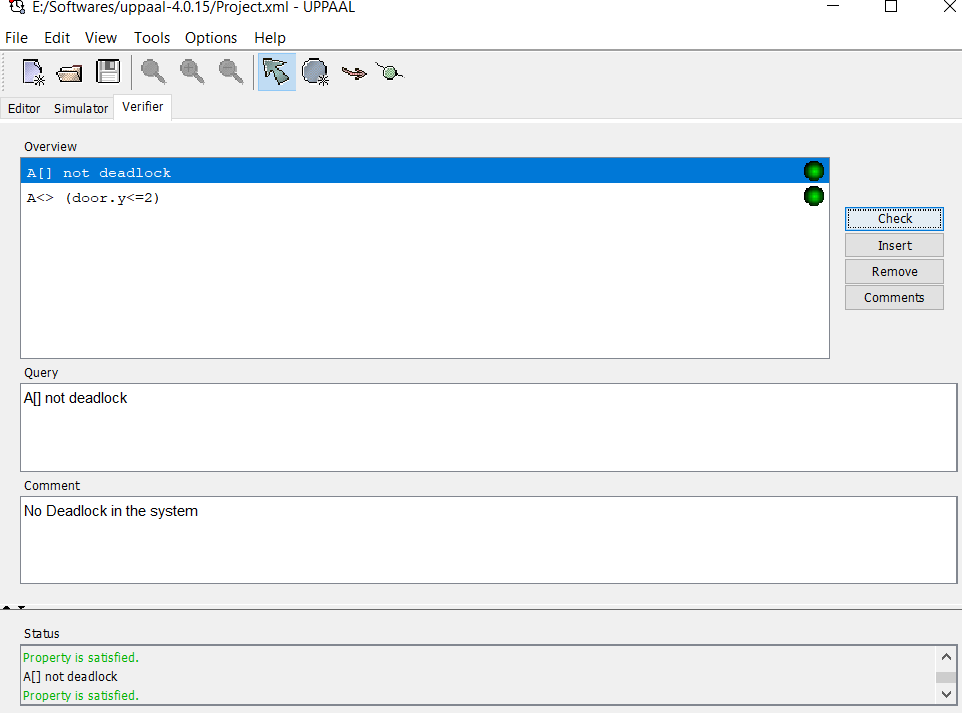
**A<> (motor.m<=2)**

****

1. **Deadlock Freeness:**

There should be no deadlock in the whole system.

**A[] not deadlock**



**7. Challenges Faced**

* Following were the different challenges we faced during this project:
* We was unable to send signal a signal to two different automaton from one automata simultaneously in UPPAL and at a time only one state changed in one automata in response to a signal. In order to counter this; we made motor signal states in my door automaton in order to signal the motor to turn on when a person approaches the mat.
* We were unable to run concreate simulator at first and we were giving invariant error. We found out that we had to initialize a clock for each automaton in order to solve this problem.
* Third problem was in writing verification properties. We was unable to find some verification properties in the system such as mutual exclusion which we have although written but we think that there could be more which we am missing.
* We also faced a challenge in global declaration and local declaration of variables and to whether we should declare them locally or globally. We used a tutorial online and came to conclusion that all the variables, which are to be used as signals or channels should be declared globally and clock variables should be declared locally.
* We also had deadlock the first time I ran the model. To solve that we have to add self-loops in some states of motor and mat.