In the Name of God

Formal Verification Computer Assignment#2

Razieh Dorehgard - 810199419

Read me No.1

First of all, the main architecture of the model is the same as the previous one. I fortunately was able to reuse my Core Rebeca model so the reactive classes, their known rebecs, statevars, and message servers are the same.

How to verify the Timed Rebeca model??

Unlike the previous project, we cannot define LTL properties, instead, the verification of the system properties will be confined to each actor reaching its deadlines, as well as the assertions.

Safety Property:

1. Assertion:

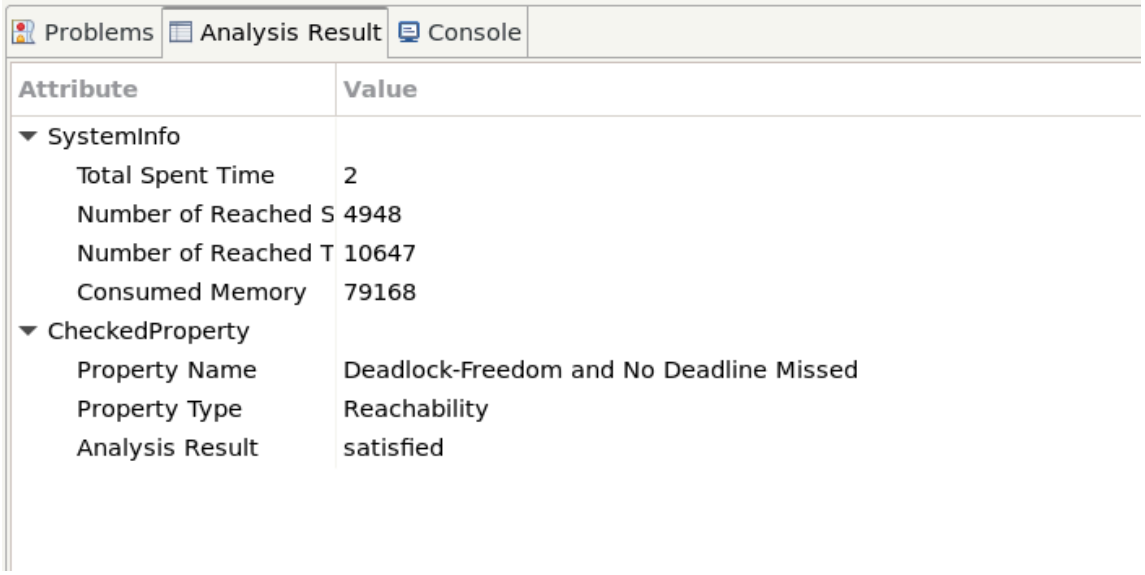
```
property {

    define {
        isSafetyViolated = ctrl.safetyCheck;
    }

    Assertion {
        Safety: !isSafetyViolated;
    }
}

```

2. The verification result:



What's new?

Based on the understanding I had from the problem and the project specification, I added some beautiful time constraints to the program:

1. I added some delay to the controller's isAllowed because the computations of the check should not take too long:

```
69    msgsrv isAllowed(boolean line, int block, int id, boolean isAuto) {
70        delay(1);
71
72        if (isAuto) {
73            autoBlock[id] = block;
74            autoLine[id] = line;
75
76            check();
77            ((Infrastructure)sender).canMove(shouldStop[id], id, true);
78            lastWasAuto = true;
79        }
80
81        if(!isAuto) {
82            mannedBlock [id] = block;
83            mannedLine [id] =line;
84
85            check();
86            ((Infrastructure)sender).canMove(shouldChangeLine[id], id, false);
87            lastWasAuto = false;
88        }
89    }
```

2. Since the infrastructure plays a role as a proxy, it will have some delay as well:

```
106    msgsrv isAllowed(boolean line, int block, int id, boolean isAuto) {
107        delay(1);
108
109        if (isAuto) {
110            autoCars[id] = ((Auto)sender);
111            ctrl.isAllowed(line, block, id, true);
112        }
113
114        if (!isAuto) {
115            mannedCars[id] = ((Manned) sender);
116            ctrl.isAllowed(line, block, id, false);
117        }
118    }
```

```
120     msgsrv canMove(boolean canMove, int id, boolean isAuto) {
121         delay(1);
122
123         if (isAuto) {
124             autoCars[id].canMove(canMove, true);
125         }
126
127         if (!isAuto) {
128             mannedCars[id].canMove(canMove, false);
129         }
130     }
```

3. When autonomous and manned cars want to ask the controller if they are allowed to move, the message shouldn't take too long to be responded.

```
firstStep = false;
infra.isAllowed(line, block, id, true) deadline(5);
asked = false;
```

4. When autonomous and manned cars want to keep moving using live() message server, the self move shouldn't take longer than 1 time unit:

```
self.live() deadline(1);
```

5. Since the movements of the manned cars might be on a higher level of nondeterminism, I decided to add more delay to the live message server of the manned reactive class; this way we have actually modelled the nondeterminism of the real world in our model and have verified that our system works correctly in those circumstances too.

```
msgsrv live() {
    delay (?(1, 2, 3));
    move();
}
```