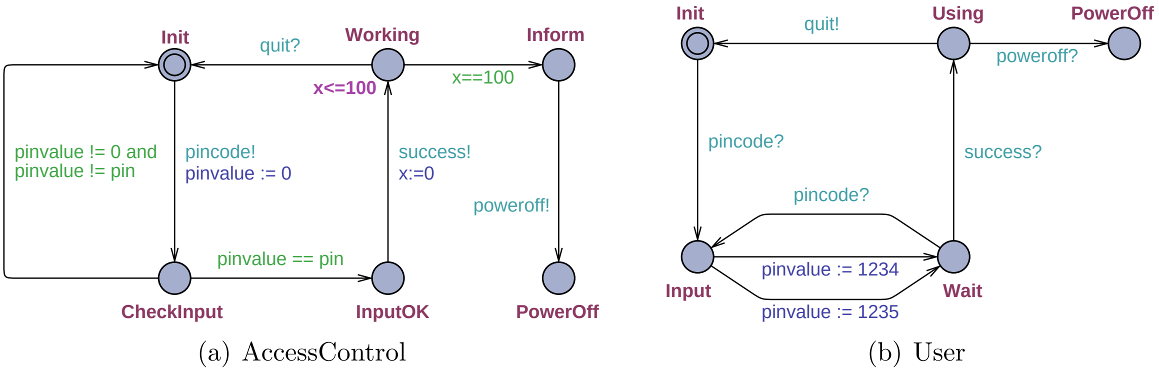
02224 Modelling and Analysis of Real-Time Systems Week 3 2022–02–16
DTU Compute Hans Henrik Løvengreen

Solution to Exercise 1: Simple Access Control

Task 1

Simulate the model in Figure 1.



(a) AccessControl (b) User

Figure 1: Version 1 of the system, `mobile1.xml` (tasks 1–4).

Task 2

The User will get access to the phone if and only if he has the correct pin-code:

```
A[] (User.Using imply pinvalue == AccessControl.pin)
```

Task 3

The User only receives a poweroff if he has not used to phone for 100 or more time units:

```
A[] (User.PowerOff imply AccessControl.x >= 100)
```

Task 4

If the AccessControl is Working, then the user is Using the phone:

```
A[] (AccessControl.Working imply User.Using)
```

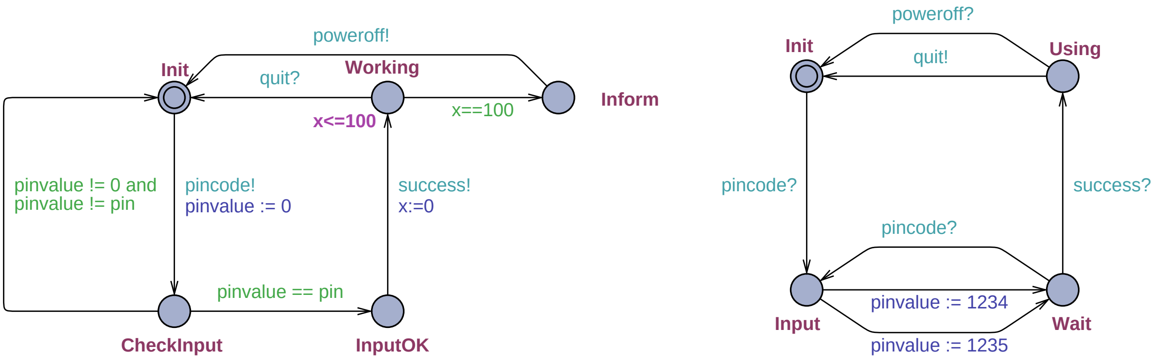
Task 5

The system in Figure 1 has a deadlock:

```
E<> (deadlock)
```

Figure 2 shows the modified system. This system has no deadlocks:

```
A[] (not deadlock)
```



(a) AccessControl

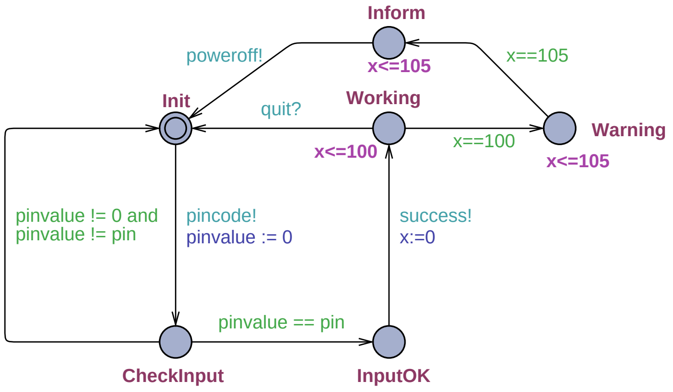
(b) User

Figure 2: Version 2 of the system, `mobile2.xml` (task 5).

Task 6

Figure 3 shows the modified system. This system gives the user 5 time-units warning time before it takes the action poweroff:

```
A[] (AccessControl.Warning imply AccessControl.x >= 100 &&
                                 AccessControl.x <= 105)
```

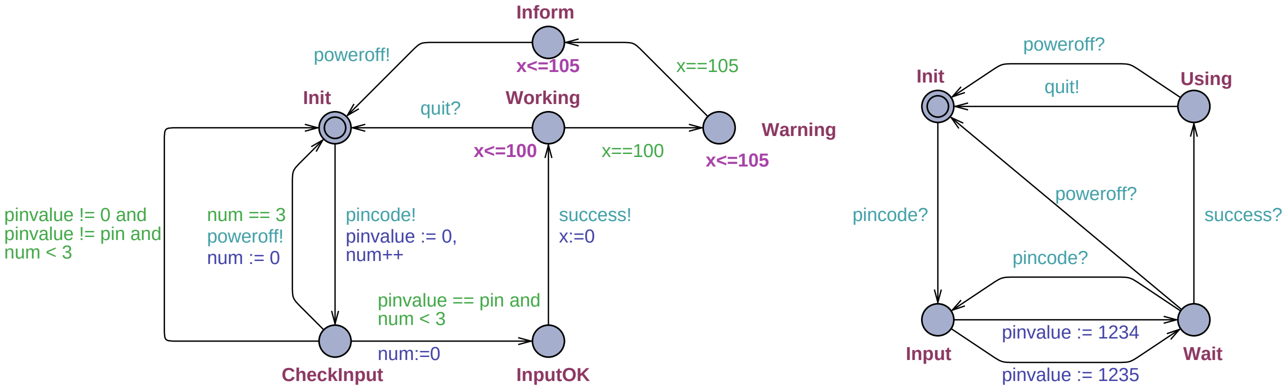


(a) AccessControl

Figure 3: Version 3 of the system, `mobile3.xml` (task 6).

Task 7

Figure 4 shows the modified system.

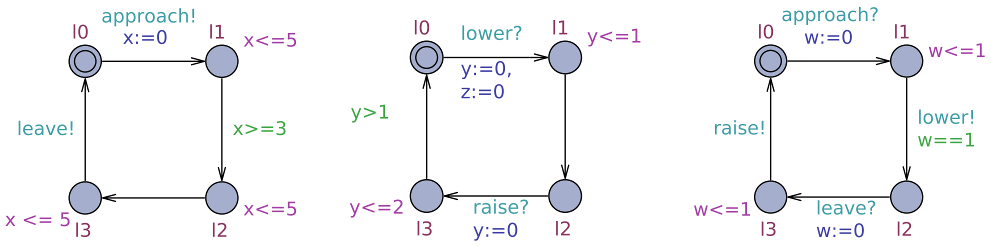


(a) AccessControl (b) User

Figure 4: Version 4 of the system, `mobile4.xml` (task 7).

Solution to Exercise 2: Simple railroad gate controller

Task 1



(a) Train (b) Gate (c) Controller

The system can be found in `train0.xml`. The clock z is used in tasks 9 and 10.

Task 2

When the train is inside the gate, the gate should be closed:

```
A[] Train.l2 imply Gate.l2
```

Task 3

Deadlock detected by:

```
E<> deadlock
```

The deadlock can be fixed by changing the guard on the edge from l_3 to l_0 in the Gate from $y > 1$ to $y \geq 1$. To verify that the modified system has no deadlocks:

```
A[] not deadlock
```

The corrected system is given as `train1.xml`.

Task 4

The train can approach the gate:

```
E<> Train.l1
```

The train can be in the gate:

```
E<> Train.l2
```

The train can exit the gate:

```
E<> Train.l3
```

Task 5

The gate can be lowered:

```
E<> Gate.l3
```

The gate can be raised:

```
E<> Gate.l1
```

Task 6

The controller can lower the gate:

```
E<> Controller.l2
```

The controller can raise the gate:

```
E<> Controller.l0
```

Task 7

Whenever the train approaches the gate, it will inevitably cross it:

```
Train.l1 --> Train.l3
```

Task 8

Whenever the gate is lowering, it will inevitably be raising again:

```
Gate.l1 --> Gate.l3
```

Task 9

The gate is never closed (or lowering/raising) for more than 10 minutes at a time:

```
Gate.l1 --> (Gate.l0 && Gate.z <= 10)
```

Task 10

The gate is closed for at least 3 minutes at a time:

```
Gate.l1 --> (Gate.l0 && Gate.z >= 3)
```

and at most 7 minutes at a time:

```
Gate.l1 --> (Gate.l0 && Gate.z <= 7)
```