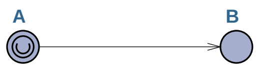
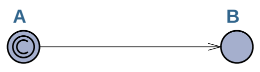
Solution to Exercise 1

Commitment and urgency

The difference between urgent and comitted states can be shown by:



(a) UrgentLoc



(b) ComittedLoc

For both of these automata, we can show that time cannot progress while in the A-state:

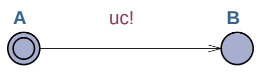
```
A[] UrgentLoc.A imply time == 0
A[] CommittedLoc.A imply time == 0
```

However, the committed state is stronger since the the urgent state cannot be left while the system is in a committed state:

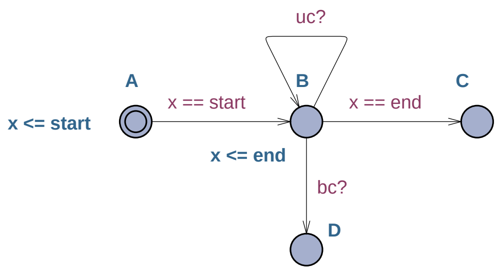
```
A[] CommittedLoc.A imply UrgentLoc.A
```

Urgent Channels

To illustrate the semantics of urgent channels, we introduce the automata:



(c) UrgentChan



(d) TimedReceiver

The latter is a template which inputs on the urgent channel `uc` in a time window between `start` and `end`. This is instantiated by:

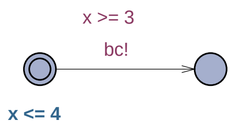
```
R1 = TimedReceiver(2,3);
R2 = TimedReceiver(2,5);
R3 = TimedReceiver(6,8);
```

The following properies verify that the communication takes places as soon as a receiver is ready:

```
A[] time < 2 imply UrgentChan.A
A[] time > 2 imply UrgentChan.B
```

Broadcast

Adding a broadcast channel `bc` and an automaton which broadcasts on this at an arbitrary moment between 3 and 4:



(e) Broadcaster

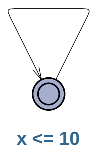
we can verify the following for the receivers R1, R2, and R3:

```
E<> R1.D
E<> not R1.D and time > 4
A[] time > 4 imply R2.D
A[] not R3.D
```

The two first properties show that R1 may or may not detect the broadast (depending on whether it leaves state B before a broadcast at time 3). The third property shows that R2 will always detect the broadcast. Finally, the fourth property shows that R3 will never see the broadcast because of disjunct time windows.

Deadlock, zeno-behaviour and time-lock

Consider the automaton:



(f) Time-lock

For this we can verify:

```
A[] !deadlock
```

In UPPAAL, a state is deadlocked if no proper transitions (ie. non-time transitions) are ever possible. Here, this holds because of the unconditional loop-transition.

To show that time-lock is not possible, we would like to show that on any path time will progress beyound some limit. For this, a global clock `time` is added and compared to a limit much larger than the timings used in the system, e.g.:

```
A<> time > 100
```

This property does not hold. It is not even possible to get to time 100 on some path as expressed by

```
E<> time > 100
```

because of the location invariant preventing time from exceeding 10.

We might then expect always to reach time 5 though, stated by:

```
A<> time > 100
```

This property, however does not hold either! The reason is here the unconditional loop, which may be taken ad infinitum withouth time progressing (e.g. at time 3). Such *zeno behaviour* also indicates a flaw in the modelling.

A UPPAAL model comprising all of the above automata and accompanying queries is found as `Week4_Ex1.xml`.