Prof. Dr. A. Podelski, Dr. B. Westphal
L. Musliu

Wintersemester 2017/18

Real-Time Systems

`http://swt.informatik.uni-freiburg.de/teaching/WS2017-18/rtsys`

Exercise Sheet 6

Early submission: Monday, 2018-01-15, 14:00 Regular submission: Tuesday, 2018-01-16, 14:00

Exercise 1 — Region Automaton ? (6/20 Points)

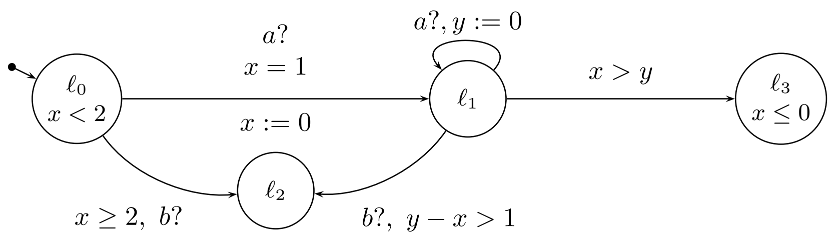

Figure 1: Timed Automaton for Exercise 1.

Consider the timed automaton $\mathcal{A}$ in Figure 1. In the tutorial, we had the impression that locations ℓ_2 and ℓ_3 are not reachable. Prove this statement by constructing the region automaton.

Hint 1: You need not give all configurations of $\mathcal{R}(\mathcal{A})$ if you explain appropriately why those, which you do show, are sufficient for the task.

Hint 2: Can you think of tools which may help you in any way to assess whether your solution is correct? If yes, you may want to use that way.

Exercise 2 — Working with Zones ? (4/20 Points)

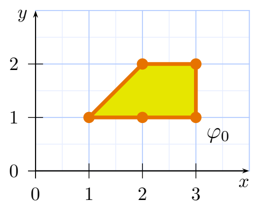

Figure 2: Zone φ_0 for Exercise 3.

(i) Compute

$$\text{Post}_e(\ell_0, z)$$

for the zone φ_0 given by Figure 2 and for both edges originating at location ℓ_0 of the timed automaton from Figure 1; give the intermediate steps up to φ_5.

In addition, give the valuation-diagram (like Figure 2) of φ_5 for each case. (3)

Hint: we want to see that the algorithm from the lecture is applied, yet you may simplify the constraints along the way. Then both, the result of applying the algorithm to obtain intermediate constraints needs to be given as well as the simplification.

(ii) What can you conclude from your answer to Task (i) about the reachability of ℓ_2? (1)

Exercise 3 — Getting Serious (10/20 Points + 5 Bonus)

The appendix describes a model of a simplified self-monitoring protocol for a wireless fire alarm system. To model message loss due to frequency jamming, there are dedicated automata for the four channels A, B, C, and D and a global variable

```
int[-1,3] gBlockedChannel;
```

whose value models the currently jammed frequency (-1: all channels free, 0: channel A jammed, …, 3: channel D jammed).

(i) The model lacks a jamming device. Extend your model of the jamming device such that the value of `gBlockedCh` is updated properly and add your resulting automaton to the model. (1)

(ii) Use Uppaal's query language to formalise and check the following properties in the model from Task (i):

a) None of the media is in location *transmit* if the sensor is in location *idle*. (1)

b) The master may correctly display a failed sensor. (1)

c) It is possible that the master displays a failed sensor while the sensor is not in location *fail*. (1)

d) There is a run where the master does not display a spurious warning. (1)

e) Sensor failure (e.g., due to low battery) is always finally detected by the master. (1)

For each property, write down what you expect as outcome and discuss why a designer would check properties of this kind for a design model (in a broader scope of general design models, not just the specific fire alarm system considered here).

f) Assume the regulation for the modelled fire alarm system requires that a sensor failure is detected 10 s after the failure the latest. Formalise this requirement (possibly after extending the model by auxiliary model elements[1]) and check it. Is the result plausible? Explain! (5 Bonus)

(iii) Assume your job is to analyse the given self-monitoring protocol for robustness against frequency jamming as considered in the EN 54-25. The self-monitoring protocol is to be used in a new fire alarm system for which no hard- or software exists yet. Further assume that your boss would favour a "more traditional" approach without any modelling efforts, so your means would be limited to:

- building prototype hardware (budget is limited to one sensor and one master),
- programming prototype software,
- testing the prototype system,
- external devices to "sniff" activity on the frequency bands may be in budget if you say it must be, but they are not easy to operate (so incur serious extra cost),
- etc.

You can assume that an implementation of a jamming device is available; for simplicity it is deterministic, that is, it jams A, B, C, and D in this order and each frequency for exactly 1 second (plus maybe a small epsilon). It keeps the frequencies free for as short as possible durations. This was considered to be the most challenging jamming behaviour which could be implemented in budget. An automaton of this deterministic jamming device is included in the model described in the appendix.

Discuss the advantages and disadvantages of applying formal modelling and analysis to your job from a Software Engineering perspective as a basis for a meeting with your boss to decide on with which method to proceed. (4)

Hint: in particular discuss costs (based on your estimations) and risks; in particular elaborate on reproducibility (recall that there may be other users of the frequency bands A to D other than your fire alarm system). You may want to in particular discuss good- and bad-case costs, where in a good case, the protocol design is correct (or has easy to fix issues) and in a bad case the protocol design is broken far beyond easy repair.

As a starting point for your discussion, you may want to first analyse whether the proposed protocol is *robust against the deterministic jamming device (by using template '*NDJammer*' and by re-checking the queries from Task (ii) on the changed model) and if not, discuss how you see the chances to detect that using only tests.*

[1] a model element, like location, edge, clock, etc. is called auxiliary if it does not change the behaviour of the design idea, but just makes conditions explicitly observable which are part of the design behaviour but otherwise only implicitly observable

Appendix — A Simple Self-Monitoring Protocol

The basic design idea for the self-monitoring protocol is to have components called *master* which are responsible for monitoring smoke/heat *sensors*. Each sensor is assigned to exactly one master and periodically send *LZ* ("I'm alive") messages at fixed points in time. If the message is not received at the expected time, the master warns about a possible sensor failure.
To avoid message collisions, a TDMA (time division multiple access) scheme is employed. That is, time is partitioned into *cycles* where each cycle is further partitioned into *windows* of equal length. Each sensor is assigned a unique window, this is where the master expects the *LZ* message.

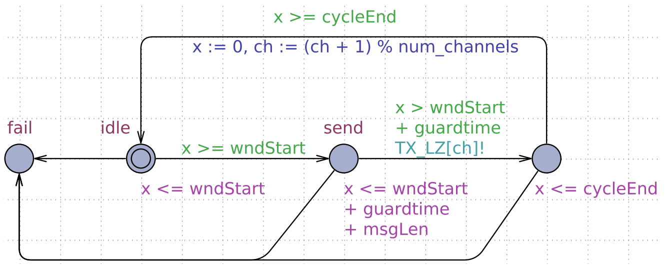

Figure 3: Model of sensor behaviour.

The model of sensor behaviour is shown in Figure 3. A sensor is idle up to the time of its window. Then it waits for a so-called *guard time* before it sends the *LZ* message (to compensate for slight differences between the local clocks in the devices in the system). *LZ* messages can be of different length up to `msgLen` time units (thus the model over-approximates, since real messages do not have arbitrarily short duration). After sending the *LZ*, the sensor waits for the end of the cycle and resets its local clock x to prepare itself for the next cycle. A sensor may spontaneously fail at any possible point in time.

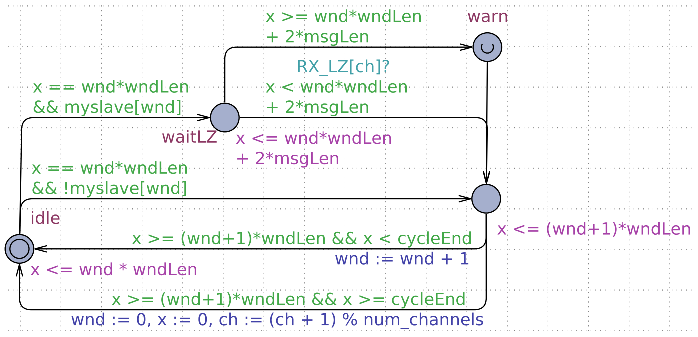

Figure 4: Model of master behaviour.

The model of master behaviour is shown in Figure 4. A master is idle at the beginning of a window and decides whether the current window is a window of one of its slaves (encoded in the boolean array `mySlave`). If this window is not one of one of this master's sensors it just waits for the next window. Otherwise it becomes ready to receive the *LZ* in location *waitLZ*. If the message is not received in time, a warning is displayed (modelled by location *warn*). At the end of a window, there are two cases to distinguish: if the last window was the last window in the cycle, a new cycle needs to be started (the number of the current window `wnd` is updated accordingly).
Note that, in order to obtain robustness against disturbances of the used frequencies, a channel rotation scheme is employed. At the end of each cycle, the next frequency band is chosen (starting with A, then B, etc. and after D going back to A). Master and sensor implement the same rotation scheme and are hence always on the same frequency.
The effect of jamming a frequency is modelled by introducing dedicated automata for the employed frequency bands (cf. Figurefig:medium). A *medium* takes the *LZ* message from the sensor and forwards it to the master if and only if its frequency is not jammed at the moment (as indicated by the value of the global variable `gBlockedCh`).

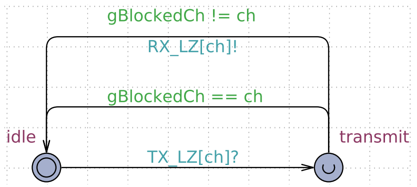

Figure 5: Environment model: communication medium.

In reality, jamming of a frequency may start or stop in the middle of a phase where a message is actually sent. In the model, this effect is visible since `TX`/`RX` actually models the point in time where message transmission is completed. For each case of jamming (starting or stopping in the middle of a transmission), there is a run of the model where the sender chooses the length of the message such that the end of the message coincides with the frequency being jammed. A robust system needs to withstand *in all cases*, in particular in those where message lengths are chosen to meet the jamming.

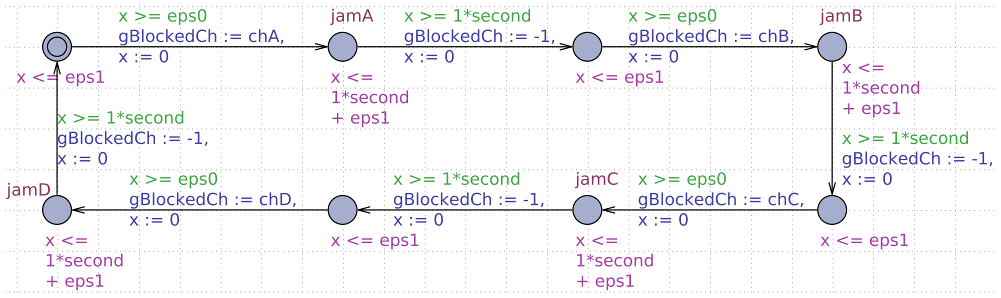

Figure 6: Model of the behaviour of the deterministic jamming device.

A model of the deterministic[2] jamming device is shown in Figure 6. The "cycle time" of the jamming system has been estimated to lie been `eps0`= 1 and `eps1`= 5 time units. There is no exact value since the CPU of the jamming system employs modern features like caches, out-of-order execution, pipelining etc. which may cause varying (and hard to predict) cycle times over time.

[2]in so far, as the sequence of blocked frequencies is deterministic, so we can imagine a deterministic program which implements this behaviour; the timing behaviour of the system (that is, the implementation running on some hardware platform) may in detail still (and of course) be highly non-deterministic in its timing behaviour