„Model checking“

Prof. Dr. Marcel Kyas

Assignment 5, November 10, 2011

Freie Universität Berlin

Exercise 12 (20 Points) Provide an example for a regular safety property P_{safe} over AP and an NFA $\mathcal{A}$ for its minimal bad prefixes such that $\mathcal{L}_\omega(\mathcal{A}) \neq (2^{AP})^\omega \setminus P_{safe}$ when $\mathcal{A}$ is viewed as an NBA.

Exercise 13 (20 Points) Show that the class of languages that are accepted by DBAs is not closed under complementation.

Exercise 14 (10 Points) Consider the GNBA outlined in Figure 1 with acceptance sets $F_1 = \{q_1\}$ and $F_2 = \{q_2\}$. Construct an equivalent NBA.

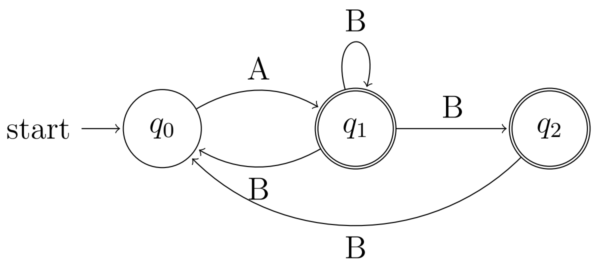

Figure 1: GNBA for Exercise 14

Exercise 15 (20 Points) Consider the following classic problem:

> A farmer has a goat, a wolf and a cabbage. He must cross a river in a boat that will only carry himself and one item at a time. If he leaves the goat and the cabbage and takes the wolf, the goat will eat the cabbage. If he leaves the wolf and the goat and takes the cabbage, the wolf will eat the goat. The same applies once they are on the other side.

Model this problem in Promela and write a specification in SPIN that solves this problem. Your claim should provide an error trail that represents a correct solution to this problem.

Exercise 16 (20 Points) This exercise deals with a simple fault-tolerant communication protocol in which processes can fail. A failed process is still able to communicate, i.e., it is able to send and receive messages, but the content of the messages it sends is unreliable. More precisely, a failed process can send messages with arbitrary content.

When we are given N reliable processes (i.e., processes that have not failed and that are working as expected) and K unreliable processes, where $N > 3K$ and $K \geq 0$. There is no way, a priori, to distinguish the reliable and the unreliable processes. All processes communicate by means of exchanging messages. Each process has a local variable, which initially has the value 0 or 1. The following informally described protocol is aimed to be followed by the reliable processes, such that at the end of $K + 1$ we have:

- eventually every reliable process has the same value in its local variable, and
- if all reliable processes have the same initial value, then their final value is the same as their common initial value.

The difficulty is to establish these constraints in the presence of K unreliable processes!

The following protocol is due to Berman and Garay [1].

Let the processes be numbered 1 through $N + K$. Processes communicate with each other in rounds. Each rounds consists of two phases of message transmission: In round i in the first phase every process sends its value to all processes, including itself; in the second phase, process i sends the majority value it received in the first phase (for the majority value to be well-defined, assume that $K + N$ is odd) to all processes. If a process receives at least N instances of the same value in its first phase of the round, it also sets its local variable to this value; otherwise it sets its local variable to the value received (from process i) in the second phase of this round.

(a) Model this protocol in Promela. Make the protocol description modular such that the number of reliable and unreliable processes can be changed easily. As the state space of your model could be very large, instantiate your model with a single unreliable process and four reliable processes.

First hint: One of the main causes for the large state space is the model for the unreliable process, so try to keep this model as simple as possible. This can be achieved by, for instance, assuming that an unreliable process can only transmit arbitrary 0 or 1 values (and not any other value) and that a process always starts with a fixed initial value (and not with a randomly chosen one). In addition, use atomic broadcast for message transmission.

Second hint: It might be convenient, but not necessary, to use a matrix of size $(N + K) \times (N + K)$ of channels for the communication structure. As Promela does not support multi-dimensional arrays, you could use the following construct (where M equals $N + K$):

```
typedef Arraychan {
  chan ch[M] = [1] of {bit}; /* M channels of size 1 */
}

Arraychan A[M]; /* A matrix A of MxM channels of size 1 */
```

Statement A[i].ch[j]!0 denotes an output of value 0 over the channel directed from i to j. Similarly, statement A[i].ch[j]?b denotes receiving a bit via the channel directed from process i to j and storing the value in b.

(b) Formalise the two constraints of the protocol in LTL and convert these into never claims.

(c) Check the two temporal logic properties by SPIN and hand in the verification output generated by SPIN.

(d) Show that the requirement $N > 3K$ is essential. What happens when you change the configuration of your model such that $N \leq 3K$ and check first of the properties. Create the shortest counter example and perform a guided simulation of this undesired scenario. Hand in the counter example you found and give an explanation of it.

References

[1] G. Berman and Juan A. Garay. Asymptotically optimal distributed consensus (extended abstract). In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *ICALP*, volume 372 of *Lecture Notes in Computer Science*, pages 80–94, Heidelberg, 1989. Springer-Verlag. `doi:10.1007/BFb0035753`.