Technische Universität München
Institut für Informatik
Prof. Dr. Helmut Veith
Dipl.-Inf. Michael Tautschnig

SS 2007
Übungsblatt 6
Mittwoch, 13. Juni 2007

Übung zur Vorlesung Model Checking

SPIN

Use SPIN to solve the following exercises. SPIN can be obtained from `http://spinroot.com`

(a) *Unreliable Channel.* Model an unreliable channel, i.e., a channel which might lose packets. Use this channel to transmit packets from a sender to a receiver process. Use SPIN to simulate the system.

(b) *Unreliable Channel: Proof.* Prove that the channel is unreliable, i.e., that not every packet that is sent, will arrive at the receiver.

(c) *Dining Philosophers.* Model the Dining Philosophers Problem in Promela with 5 philosophers. Prove that they could starve. Modify the model such that each philosopher will eventually eat. Prove this property.

(d) * *Token Ring.* A token ring consists of m independent processors which are arranged in a cycle, where each processor is connected to its left and right neighbors. The processes of the token ring use a token (represented by a message in channel) to synchronize each other. After each processing step, the token is passed on to one of its neighbors.

- Implement the token ring for $m = 4$, where the token is passed to one of the neighbors nondeterministically. Simulate the token ring in the interactive environment.
- Use SPIN to check whether it is guaranteed that at most one processor gets access to the critical section at the same time.
- Use SPIN to check whether a deadlock can occur.
- Use SPIN to check whether at least one process enters the critical section infinitely often, i.e., whether progress is achieved.
- Repeat the above steps for a model where the token is passed deterministically to the left.
- *Optional:* Find the maximal m for which you can verify the model on your machine.

(e) * *Token Ring: Fairness.* Use the two models from the last exercise. Use SPIN to produce a `never` claim which states that each process gets access to the critical section infinitely often, i.e., whether each process is able to make progress. Check this condition.

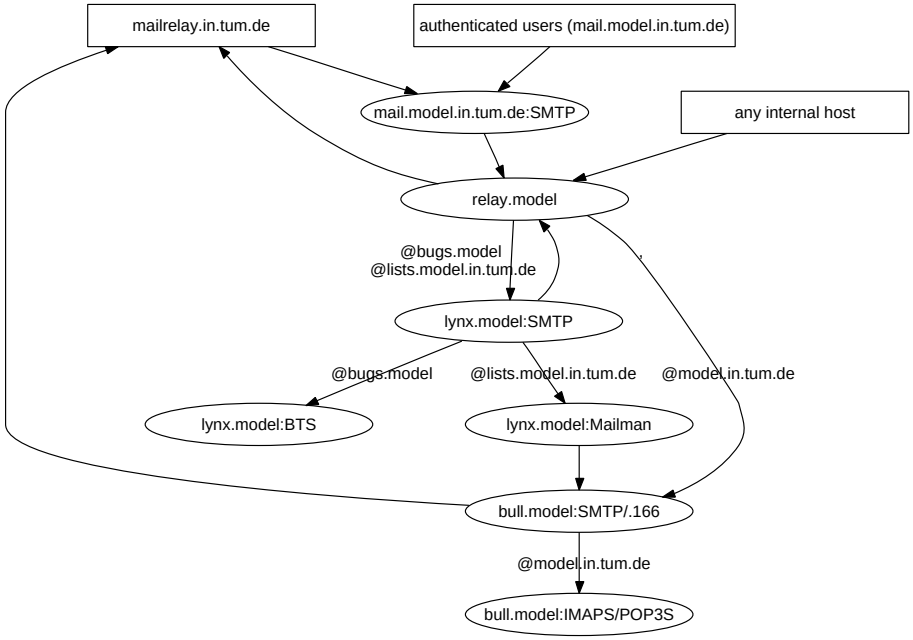

Figure 1: An email delivery system.

(f) * *Mail system.* Model the mail system shown in Figure 1 in Promela using proper abstractions from the irrelevant details.

The following assumptions are made: (1) Messages are only generated by the nodes drawn as boxes. (2) Edges marked with domains only apply to emails with such recipients, edges without domains apply to all remaining domains. (3) No email loops through mailrelay.in.tum.de.

Using SPIN to verify that

i) no message loops forever

ii) every message passes through relay.model at least once

iii) the maximum number of hops is smaller than ten – is it also smaller than 5?