Stefan Rührup
Computer Networks and Telematics
University of Freiburg, Germany

Network Protocol Design
and Evaluation
Summer 2009

Exercise No. 6

June 19, 2009

Task 1 *The Pathfinder Problem*

You are chief engineer at JPL and responsible for the Mars Pathfinder Mission. After the spacecraft has landed and released the rover, it is expected to transmit data to the earth. Unfortunately, the contact to the craft is lost at unpredictable moments. You suspect an automatic software reset after a process is blocked. There is a process for gathering meteorological data (low priority) and another process that consumes data (high priority). Both access an internal bus an use a semaphore that restricts the access. The following Promela model shows the mechanism.

```
mtype = { free, busy, idle, waiting, running };

show mtype h_state = idle;
show mtype l_state = idle;
show mtype mutex = free;

active proctype high() /* can run at any time */
{
end: do
     :: h_state = waiting;
         atomic { mutex == free -> mutex = busy };
         h_state = running;

         /* critical section - consume data */

         atomic { h_state = idle; mutex = free }
    od
}

active proctype low() provided (h_state == idle)
{                           /* scheduling rule */
end: do
     :: l_state = waiting;
         atomic { mutex == free -> mutex = busy};
         l_state = running;

         /* critical section - produce data */

         atomic { l_state = idle; mutex = free }
     od
}
```

You can find the Promela model in the examples directory of the SPIN distribution.

The problem description and the Promela model appeared in G. J. Holzmann, "Designing Executable Abstractions", 2nd Workshop on Formal Methods in Software Practice, 1998, pp. 103–108.

1. Analyze the model with SPIN. What is the problem?
2. Describe the processes in form of automata. Derive the complete state space for the two processes and the mutex state.
3. How can the problem be solved?

Task 2 *LTL and Never claims*

1. Specify the recurrence of a property p in LTL. Describe the Büchi automaton for the negated formula and give the corresponding never claim.
2. You want to check an invariant property p, but you have already used the never claim. You define another process by

   ```
   active proctype invariant() { do :: assert(p) od }
   ```

   What is the problem of this solution (hint: think of timeouts)? Is there an alternative?