Stefan Rührup
Computer Networks and Telematics
University of Freiburg, Germany

Exercise No. 5

June 12, 2009

Task 1 *Dijkstra's Semaphore*
Consider the following Promela model for the Semaphore (cf. Chapter 5.2 of the lecture).

```
mtype {p,v}

chan sema = [0] of {mtype}

active proctype Semaphore() {
end:        do
            :: sema!p ->
progress:      sema?v
            od
}

active [3] proctype user() {
           do
           :: sema?p;  /* enter critical section */
critical:     skip;    /* critical section */
              sema!v;  /* leave critical section */
           od
}
```

1. Warm-up:
   (a) Make yourself familiar with SPIN. Run a simulation and generate a sequence chart using XSPIN or spin with the command line parameters -c or -M.
   (b) Build a verifier and compile it using the cc parameters -DNOREDUCE and -DNP
   (c) Check for non-progress cycles.
2. Insert a correctness claim stating that at most one process can enter its critical section at any time. What do you use, assertions, meta-labels or a never-claim? Check the correctness with SPIN.
3. Extend the semaphore such that an arbitrary fixed number of processes can enter their critical section at any time.
4. Implement a semaphore without a semaphore process by using only a channel. Validate your model.

Task 2 *Protocol Validation*
Consider the Promela model for the Alternating Bit Protocol, which comes with this exercise sheet, and an erroneous channel (cf. Chapter 5.2 of the lecture).

1. Change the Promela model of the lower layer such that messages might get lost.
2. Extend the communication protocol such that messages are retransmitted after they were lost. Simulate and verify your protocol.
3. Extend your protocol such that it also works when messages are delivered out of order. Introduce sequence numbers (Exercise 3, Task 2) and extend the channel model. Simulate and verify your protocol