



A. Podelski
A. Malkis
M. Wehrle
M. Mehlmann

November 29, 2007
Discussion: Week of Dec 3 – Dec 7
Different Rooms

Tutorials for Verification Exercise sheet 5 (Spin)

Exercise 1: Bakery mutual exclusion

Consider the following Promela model implementing the so-called bakery mutual exclusion protocol.

```
byte turn_A; /* turn_A = 0 iff A does not want to enter CS */
byte turn_B; /* turn_A = 0 iff A does not want to enter CS */

proctype A() {
    think: skip;
    req:   turn_A = 1;
          turn_A = turn_B + 1;
    wait:  turn_B == 0 || turn_A < turn_B;
    crit:  skip; /* critical section */
    leave: turn_A = 0;
    goto think
}

proctype B() {
    think: skip;
    req:   turn_B = 1;
          turn_B = turn_A + 1;
    wait:  turn_A == 0 || turn_B < turn_A;
    crit:  skip; /* critical section */
    leave: turn_B = 0;
    goto think
}

init { atomic { run A(); run B(); } }
```

- Express the properties *mutual exclusion* (never both processes at label use), *response* (if a process is at wait then it will proceed to use eventually), and *no overtaking* (if a process is waiting to enter the critical section then the other process cannot enter the critical section twice).
- Verify the above properties using SPIN.
- Why does mutual exclusion not hold? Express the reason as a formula and verify it.
Hint: The formula only involves the turn variables.
- Prove that mutual exclusion holds if the protocol does not run too long.

Exercise 2: Nested Peterson's mutual exclusion

Recall Peterson's mutual exclusion protocol from the second exercises sheet (and from the lecture).

- Generalize the protocol from 2 to 4 processes.
Hint: Nest multiple instances of the 2-process protocol.
- Express the *mutual exclusion* property and verify it using SPIN.
- Express the *non-starvation* property and verify it using SPIN. You may need fairness assumptions.

Exercise 3: The web shop revisited

Consider the web shop from the second exercise sheet.

- (a) Give an implementation for the solution in Exercise 5 (a) in Promela. Use byte fields to represent customers and products.
- (b) To simulate the web shop, you need an implementation for the environment, in this case you need some customers ordering products and the Schufa. Include the following processes into your promela program.

```
bool creditworthy[7];

active proctype Customer0()
{
    byte p, verp;
    if :: creditworthy[0] = 1 :: creditworthy[0] = 0 fi;
    do :: true ->
        if :: p = 1 :: p = 2 fi;
        order!0,p;
        if
            :: deliver?0,verp -> assert (creditworthy[0] && p == verp)
            :: denial?0 -> assert (!creditworthy[0])
        fi
    od
}

active proctype Customer1()
{
    byte p, verp;
    if :: creditworthy[1] = 1 :: creditworthy[1] = 0 fi;
    do :: true ->
        if :: p = 1 :: p = 2 fi;
        order!1,p;
        if
            :: deliver?1,verp -> assert (creditworthy[1] && p == verp)
            :: denial?1 -> assert (!creditworthy[1])
        fi
    od
}

active proctype Schufa()
{
    byte c;
    do
        :: check?c -> reply!creditworthy[c]
    od
}
```

You can now simulate your web shop or check safety properties.

- (c) Give an implementation for the refined web shop of exercise 5 (b) on the second exercise sheet. Again, simulate and check the safety of the web shop
- (d) Add some progress labels and check whether each customer does eventually get his order delivered if he is creditworthy. Add fairness constraints if necessary.
- (e) Try to add more customers, more products and try to increase the capacity of the queues. How many can the model-checker handle with reasonable memory size?