„Model checking“
Prof. Dr. Marcel Kyas
Assignment 2, October 14, 2009

Freie Universität Berlin

Exercise 3 (2 Points) We are given three processes P_1, P_2 and P_3 that share the integer variable x. The program that is given for each process P_i ($i \in \{1, 2, 3\}$ is:

for (k_i=1;k_i<=10;++k_i) {
  int $r_i = x$;
  r_i++;
  x=r_i;
}

That is, P_i executes ten times the assignment $x = x + 1$. This assignment is realised by loading the shared variable into a local register, increasing the value of that register and storing the local register into the shared variable. Consider the parallel program P:

x:=0;
$P_1 \parallel P_2 \parallel P_3$;

Does P have an execution that halts in a final state with the value $x = 2$?

Exercise 4 (6 Points) The following program is a mutual exclusion protocol for two processes due to Amir Pnueli. There is a single shared variable s that is either 0 or 1 and initially 1. Besides, each process has a local Boolean variable y that initially equals 0. The program text for process P_i ($i \in \{0, 1\}$ is as follows:

for(;;) {
  // Non−critical section
  (y_i,s) = (1,i);
  await ((y_{i-1}==0) || (s != i));
  // critical section
  y_i = 0;
}

Here, the statement (y_i,s)=(1,i) is a multiple assignment in which y_i=1 and s=i are executed as one single atomic step.

1. Define the program graph of a process in Pnueli's algorithm.
2. Determine the transition system for each process.
3. Construct their parallel composition.
4. Check, whether the algorithm ensures mutual exclusion.
5. Check, whether the algorithm ensures absence of deadlock.
6. Check, whether the algorithm ensures starvation freedom.

The last three questions may be answered by inspecting the transition system.

Exercise 5 (4 Points) The following incorrect mutual exclusion algorithm has been published in the January 1966 issue of the „Communication of the ACM“. The algorithm is for two processes; let $i \in \{0, 1\}$ be their identities. It uses three shared variables turn, flag[0] and flag[1]. Initially, flag[0]=0 and flag[1]=0. The initial value of turn is either 0 or 1.

```
process P[i = 0,1] {
  for (;;) {
    // Remainder
    flag[i] = 1;
    while (turn == 1 − i) {
      await flag[1−i] == 0;
      turn = i;
    }
    // Critical section
    flag[i] = 0;
  }
}
```

1. Formalise this algorithm in Promela
2. Augment the program such that we can identify the error in the program
3. Use SPIN to find the error in this algorithm
4. Use the counter example generated by SPIN to explain the error in the program

Handing in this Assignment Please submit your hand-written solutions on paper no later than October 28, 2009, 18:00 (before the tutorial session).