Automated Verification

Assignment 3

(Due on Tue 21 March 2006)

1. Implement the mutual exclusion algo below (the one we have discussed in class) in Promela. Formulate a safety property (the two processes are never in their critical section simultaneously) and a liveness property (if process 0 "wants" he eventually gets to enter his critical section). Verify these properties in Spin, and explain the counter example whenever there is one.

   Initially turn is 0, and want0, want1 are false;

```
process0 {
   do forever {
        want0 := true;
        while (want1 && turn == 1) {
              /* do nothing */
        }
        turn := 0;
        enter CS;
        want0 :=  false;
        turn := 1;
   }
}
```

   And symmetrically for process1.

2. Implement Peterson's algorithm for mutual exclusion in SPIN (the algorithm is a variant of the one above, where each process first sets turn to the other process (after setting his own 'want' flag) and then waits till either the other process does not 'want' or the turn points back to him.

   Check if mutual exclusion and starvation freedom are met.

3. Here is a problem from a list of exercises given on the Spin website (http://www.spinroot.com/spin/Man/Exercises.html).

   If two or more concurrent processes execute the same code and access the same data, there is a potential problem that they may overwrite each others results and corrupt the data. The mutual exclusion problem is the problem of restricting access to a critical section in the code to a single process at a time, assuming only the indivisibility of read and write instructions. (The problem disappears if one can assume an indivisible test-and-set instruction.) The problem and a first solution were first published by Dijkstra.

The following 'improved' solution appeared one year later in the same journal (Comm. of the ACM, Vol. 9, No. 1, p. 45) by another author. It is reproduced here as it was published (in pseudo Algol).

```
 1 Boolean array b(0;1) integer k, i, j,
 2 comment process i, with i either 0 or 1 and j = 1-i;
 3 C0: b(i) := false;
 4 C1: if k != i then begin
 5 C2: if not b(j) then go to C2;
 6     else k := i; go to C1 end;
 7     else critical section;
 8     b(i) := true;
 9     remainder of program;
10     go to C0;
11     end
```

Model the solution in Promela, and prove or disprove the correctness of the algorithm.

4. This question involves reasoning about the correctness of the Alternating Bit Protocol using Spin. For more details on the protocol look at Chapter 4 of Holzmann's book (Design and Validation of Computer Protocols) available online at www.spinroot.com and "spin-book-ch4.pdf" on the course webpage.

   Formulate an appropriate correctness criterion for the protocol. Verify it in Spin. Also verify whether the protocol is robust (i.e. it satisfies the correctness criterion above) in the face of messages being (a) lost, (b) duplicated, (c) and reordered.