T–79.4301 **Spring 2006**
Parallel and Distributed Systems
Tutorial 2 - Wed Feb 8, 2006 11:15 and Fri Feb 10, 2006 14:15

1. Hyman suggested in 1966 the following algorithm to solve the mutual exclusion problem. Use a maximum of 10 minutes of time and see whether you can in that time manually analyse whether it guarantees mutual exclusion:

```
// Hyman's algorithm, two parallel processes 0 and 1

var want: array[0..1] of boolean;  // want[0..1] initially false
var turn: int;                     // turn initialised to 0

// Pseudocode for process i

process P():
     var i, j: int;  // i is my index, j is the other process
     i := mypid(); j := 1-mypid();
     while true do

          // [noncritical section]

          want[i] := true;
          // [trying section]
          while true do
               if(turn != i) then begin
                    while true do
                         if(!want[j]) then begin
                              turn := i;
                              break;
                         endif;
                    enddo;
               end;
               else begin

                    // [critical section]

                    want[i] := false;
                    break;
               end;
          enddo;
     enddo;
endprocess;
```

2. Demo exercise: Model Hyman's algorithm in Promela, and use the Spin model checker to check whether it is correct.

3. The following buggy mutex algorithm was suggested by a major computer manufacturer. Use a maximum of 10 minutes of time and see whether you can in that time manually find an execution which leads to both processes being in the critical section at the same time. (Don't worry too much if you have to give up.)

```
// Buggy mutex algorithm, two parallel processes 0 and 1
var x: int; // x initialised to 0
var y: int; // y initialised to 0
var z: int; // z initialised to 0

// Pseudocode for process i
process P():
     var me: int;
     me := mypid()+1; // me is either 1 or 2
     while true do
          // [noncritical section]
          while true do
               // [trying section]
               x := me;
               // Assume atomic execution of (y != 0 and y != me)
               if(y != 0 and y != me) then
                    continue;
               endif;

               z := me;
               if(x != me) then
                    continue;
               endif;

               y := me;
               if(z != me) then
                    continue;
               endif;

               // [critical section]
               break;
          enddo;
     enddo;
endprocess;
```