Mutual Exclusion

Alessandro Artale
Faculty of Computer Science – Free Univeristy of Bolzano
artale@inf.unibz.it

You are asked to write a NuSMV program that models a three-processes asynchronous mutual exclusion protocol. Each process has a status that could be "NonCritical", "Trying" or "Critical". The rule to access the resource is based on a FIFO queue: each process's request to access the resource is stored in a "Waiting List", the first process in this list is the one which can access first the resource. The following is an un-complete NuSMV program:

```
MODULE main

VAR

-- ‘‘Wait_List_i’’ captures the ‘‘i’’ position in the queue.
-- - The values of each position are:
-- - 0: No process is stored for that position;
-- - 1: Process ‘‘1’’ is stored;
-- - 2: Process ‘‘2’’ is stored;
-- - 3: Process ‘‘3’’ is stored.
-- - Example: Wait_List1=3, Wait_List2=1, Wait_List3=0, means:
-- - There are processes ‘‘1’’ and ‘‘3’’ waiting and
-- - process ‘‘3’’ will be the first one accessing the resource.

Wait_List1: {0,1,2,3};
Wait_List2: {0,1,2,3};
Wait_List3: {0,1,2,3};

pr1: process prc(Wait_List1,Wait_List2,Wait_List3,resource_st,1);
pr2: process prc(Wait_List1,Wait_List2,Wait_List3,resource_st,2);
```

```
pr3: process prc(Wait_List1,Wait_List2,Wait_List3,resource_st,3);

ASSIGN

init(Wait_List1) := 0;
init(Wait_List2) := 0;
init(Wait_List3) := 0;

-- ‘‘resource_st’’ is a boolean variable that is true
-- when the resource is used by one of the processes.
DEFINE

resource_st := (pr1.st = c) | (pr2.st = c) | (pr3.st = c);


-------------------------------------------------------------
---------- PROCESS MODULE -----------------------------------
-------------------------------------------------------------
MODULE prc(Wait_List1,Wait_List2,Wait_List3,resource_st,myturn)

VAR

-- Variable "st" codifies the current status of each process
-- - "n": NonCritical: outside the critical section;
-- - "t": Trying: trying to enter the critical section;
-- - "c": Critical: inside the critical section.
st: {n, t, c};

ASSIGN

init(st) := n;

next(st) := case
   (st = n)                                    : {t,n};
   (st = t) & !resource_st & Wait_List1=myturn : c;
   (st = c)                                    : {c,n};
   1                                           : st;
esac;

next(Wait_List1) := case
```

```
      ???
esac;

next(Wait_List2) := case
     ???
esac;

next(Wait_List3) := case
    ???
esac;

FAIRNESS
running;

FAIRNESS
!(st = c);
```

You need to complete the program by specifying the next states for the waiting list. You need also to add CTL/LTL specifications to guarantee the following properties:

1. Two processes cannot be at the same time in their critical section.
2. If a process tries to enter its critical section, it will eventually succeed.

Remark: You are free to change the above SMV skeleton adopting a different solution as soon as the two properties above remain true and the FIFO selection is respected.