Verification of Cyber-Physical System
Fall 2017

Exercice Sheet 2

Author : Sylvain Julmy

Professor : Ultes-Nitsche Ulrich
Assistant : Prisca Dotti

Exercice 1

The following *Promela* model is equivalent to the one given in exercice:

```
#define other ((idx+1) % 2)

byte p1,p2;

bool mut[2];
bool turn;

proctype process(int idx){
    start :
    mut[idx] = true;
    turn = idx;
    (mut[other] == false || turn == other);

    /* critical section */
    critical : skip;

    mut[idx] = false;
    goto start;
}

init {
    mut[0] = false;
    mut[1] = false;
    atomic {
    p1 = run process(0);
    p2 = run process(1)
    }
}

never{
    do
        :: process[p1]@critical && process[p2]@critical; break;
        :: else;
    od
}
```

we use a generic *Promela* process to simulate two process which want to access a critical section represented by the `critical:` label. We use a never claim too : if both process are at the `critical:` label, the never claim is allowed to end, which should not be permitted by a valid model.

Exercice 2

The mutual exclusion problem states that any of the process A and B are at the same time in the critical section. We can translate this to the LTL formula $f = \Box(\neg inCS_A \wedge \neg inCS_b)$, where $inCS_p$ denots "process p is in critical section". Because that is a behavior of the system which should never happen, we use the following spin command to generate the never claim :

```
spin -f '!( []!(p && q) )'
```

which give us the following never claim :

```
never {    /* !( []!(p && q) ) */
  T0_init:
  do
  :: atomic { ((p && q)) -> assert(!((p && q))) }
  :: (1) -> goto T0_init
  od;

  accept_all:
  skip
}
```

Finally we would have the following promela model :

```
/* LTL formula and variable */

#define p true
#define q true

/* Main program */

#define other ((idx+1) % 2)

byte p1,p2;

bool mut[2];
bool turn;

proctype process(int idx){
    start :
    mut[idx] = true;
    turn = idx;
    (mut[other] == false || turn == other);

    /* critical section */
    critical : skip;

    mut[idx] = false;
    goto start;
}

init {
    mut[0] = false;
    mut[1] = false;
```

```
    atomic {
    p1 = run process(0);
    p2 = run process(1)
    }
}

never  {    /* !( []!(p && q) ) */
T0_init:
        do
        :: atomic { ((p && q)) -> assert(!((p && q))) }
        :: (1) -> goto T0_init
        od;
accept_all:
        skip
}
```

We can also check manually the *LTL* formula by forcing p and q to *true* :

```
/* LTL formula and variable */

#define p true
#define q true
```

Which leads into an error on the verification of the *Promela* model.

Exercice 3

The *Promela* model given in exercice 3 is corresponding to the *LTL* formula $f = \Diamond\Box p$, which means, with respect to the never claim, "It should never happen that, at the moment, p will always be *true* in the future".

Its like the following sequence should never happen :

$$(1), (0), (0), (1), (0), \underbrace{(1)}_{a}, (\cdots), (1), (\cdots)$$

because at the moment a, p never change back to $false$.