

COMP4100 Lectures in 2007

by Malcolm Newey

Lecture 14 30th March, 2007

SMV by Example

References:

The SMV source distribution from CMU

<http://www.cs.cmu.edu/~modelcheck/smv.html>

The SMV guided tour

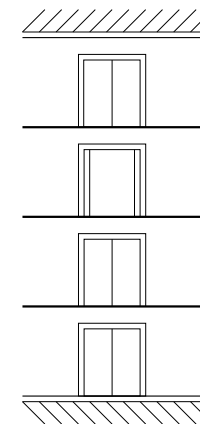
<http://www.cs.cmu.edu/~modelcheck/tour.htm>

The Lift Example

```

MODULE main
VAR floor: 0..3;
    direction: {up, down};
ASSIGN
    next(floor) := case
        direction=up & floor<3: floor+1
        direction=down & floor>0: floor-1
    1: floor
    esac
    next(direction) := case
        direction=up & floor=2: down
    1: direction
    esac

```



Points to Note

- Module `main`
- Enumerations
- boolean values are 0,1
- Types are finite
- 'control flow' in case statement
- Simultaneous assignment

There are no properties listed for SMV to check, which is just as well since the lift gets stuck eventually. It's left as an exercise to improve the model.

Semaphore Example

```

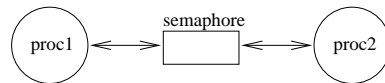
MODULE main
VAR
    semaphore : boolean;
    proc1 : process user(semaphore);
    proc2 : process user(semaphore);
ASSIGN
    init(semaphore) := 0;
SPEC
    AG (proc1.state = entering -> AF proc1.state = critical)

```

What System is being Modelled

It is a system of 2 processes which synchronize by using a shared boolean variable. The processes are active while the shared variable is passive.

The state of the shared variable is changed by exactly one of the processes making a change.



Points to Note

- Variables can be module instances
- `semaphore` is initialized in `main` but not updated there.
- `process` is a reserved word
- Parameters bound to local variable of module `main`
- In specification clause
AG is "for all traces, for all time (globally)"
AF is "for all traces, at some time"

Semaphore Example (ctd)

```

MODULE user(semaphore)
VAR
  state : {idle,entering,critical,exiting};
ASSIGN
  init(state) := idle;
  next(state) :=
    case
      state = idle : {idle,entering};
      state = entering & !semaphore : critical;
      state = critical : {critical,exiting};
      state = exiting : idle;
    1 : state;
  esac;

```

Semaphore Example (ctd)

```

next(semaphore) :=
  case
    state = entering : 1;
    state = exiting : 0;
  1 : semaphore;
  esac;
FAIRNESS running

```

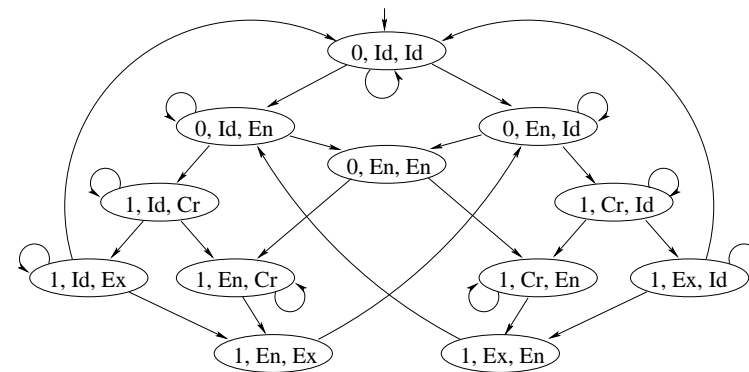
Points to Note

- In the case that `next(state)` is specified to be in the set `{idle, entering}` the choice is nondeterministic.
- Unlike in Ada, there is no nondeterminism in the selection of a clause in a case construct. The guards are examined in sequence and the first clause with a true guard specifies the new value for the variable.
- *FAIRNESS running* says module doesn't starve.
- Parameters bound to local variables of module (instance)

The Reachability Graph

System state is captured by a triple $\langle semaphore, proc1.state, proc2.state \rangle$.

The reachable states and the possible state changes are as follows:



Mutual Exclusion Example

```

MODULE main
  VAR
    s0: {noncritical, trying, critical, ready};
    s1: {noncritical, trying, critical, ready};
    turn: boolean;
    pr0: process prc(s0, s1, turn, 0);
    pr1: process prc(s1, s0, turn, 1);

  ASSIGN init(turn) := 0;

  FAIRNESS !(s0 = critical)
  FAIRNESS !(s1 = critical)
    
```

Points to Note

- Syntax for negation
- *FAIRNESS P* says *P* true infinitely often
- Parameters bound to local variables of module `main`
- s_i is state for pr_i
- 4th parameter of `prc` tells 'who am I'.

This is the example from the SMV Guided Tour

Mutual Exclusion Example (ctd)

```

MODULE prc(state0, state1, turn, turn0)
ASSIGN
init(state0) := noncritical;
next(state0) :=
case
  (state0=noncritical) : {trying,noncritical};
  (state0=trying) & ((state1=noncritical) |
    (state1=trying) | (state1=ready)): ready;
  (state0=ready): critical;
  (state0=trying) & (state1=trying) & (turn=turn0): critical;
  (state0=critical) : {critical,noncritical};
  1: state0;
esac;

```

Mutual Exclusion Example (ctd)

```

next(turn) :=
case
  turn = turn0 & state0 = critical: !turn;
  1: turn;
esac;

FAIRNESS running

SPEC AG((s0 = critical) -> !(s1 = critical))
SPEC AG((s1 = critical) -> !(s0 = critical))

SPEC AG((s0 = trying) -> AF(s0 = critical))
SPEC AG((s1 = trying) -> AF(s1 = critical))

```

Points to Note

- The same ones as before.

This model is defective. You could compute the reachability graph and discover what a model checker might report as a problem, given the requirements specified.

Practical Use

- SMV is famous.
- It is used for teaching
- It can be used for practical applications BUT
(Then again, Visual Basic can be used for practical applications.)
- SPIN is the current tool of choice for model checking