

**GS03/4204: Validation and Verification.**  
**Coursework 2: CTL and NuSMV**  
**Deadline 26 February 2007**

Franco Raimondi

<http://www.cs.ucl.ac.uk/staff/f.raimondi>

Consider the following scenario: a Sender wants to send the value of a bit (either 0 or 1) to a Receiver. The Receiver can be either idle or busy. If the Receiver is idle when it receives the bit, then the Receiver acknowledges the receipt of the bit. Otherwise, no acknowledgement is sent.

- **Task 1** Write the following code in a file and call it `btpr.smv`. Run NuSMV and perform the preliminary operations to check formulas.

```
MODULE sender(ack)
-- This module models the sender
VAR
  state : {b0,b1,recack};
  -- Just 3 states: bit0, bit1, or recack.
ASSIGN
  init(state):={b0,b1};
  -- At the beginning, the sender is in state b0 or b1 (randomly)

  -- The evolution function: if sending a bit and an ack has been
  -- received, then change state to recack. Otherwise, do not change state.
  next(state) := case
    state=b0 & ack: recack;
    state=b1 & ack: recack;
    1 : state;
  esac;
-- end of the Sender module

MODULE receiver(bit)
VAR state : {idle,busy};
    ack : boolean;
ASSIGN
  init (state) := {idle,busy};
  -- state is either idle or busy, and it changes randomly. So, no
  -- evolution function for this variable.

  init(ack):= 0;
  -- ack is initially false

  -- ack is true only if state is idle and a bit has been sent.
  next(ack):= case
    state=idle & (bit=b0 | bit=b1): 1;
    1 : 0; -- this means: in every other case ack=0.
  esac;

MODULE main
VAR
```

```
-- Create two modules, one for sender and one for receiver.  
Sen : sender(Rec.ack);  
Rec : receiver(Sen.state);
```

- **Task 2** Encode the following requirements in CTL and check them with NuSMV (you should write them in NuSMV syntax and add a line at the end of the file `btp.smv` for each of them, of the form `SPEC (your formula here)`. Check if they are true or false. NOTICE: in your answer you should provide the formulas as well as the result (true or false).

- EXAMPLE: It is always true that, if the state of the sender is `b0`, then along all paths in the future the state of the sender will be `recack`. This is encoded as:  
`SPEC AG(Sen.state=b0 -> AF(Sen.state=recack))`  
(check this one in NuSMV).
- It is always true that, if the state of the sender is `b1`, then *there exists a path* such that in the future the state of the sender will be `recack`.
- If the state of the sender is `b0`, then there is no state in the future in which the state of the sender is `b1`.
- It is always true that if the state of the receiver is `busy`, then the state of the sender will never be `recack`.

- **Task 3 (optional)** Add the following line after the declaration of the `main` module:

```
FAIRNESS !(Rec.state=busy)
```

This line forbids all executions in which the state of the Receiver is always busy.

Check all the four formulas of the previous task under this constraint and report the results.

- **Task 4** Please write a comment about the lectures on CTL and NuSMV: what is CTL? which kind of requirements can be expressed using CTL? what are the applications of model checking?

Also, feel free to add comments and feedback about this week's lectures; did you find them useful for your career?

**INSTRUCTIONS:** You should submit a report (PDF, text file, or Word) with the required answers. Also, include the SMV file for the example above with the required formulas. **The deadline is 26 February 2007.**

**Marking:** Graded A-F (C is satisfactory - a reasonable Alloy specification and written report, B-A for progressively better if optional work done, D-E for worse and unsatisfactory).