Weryfikacja wspomagana komputerowo

2016-2017

Lab 7 – NuXmv and CTL

Vincent Penelle

Topics of this Lab

- CTL verification with NuXmv
- More modelling by yourself

Exercise 1: (CTL) In addition to LTL, NuXmv is also able to check CTL formula (actually, it is its main use). CTL is a logic which deals with paths rather than a single execution. To write a CTL formula to your model, you simply write `SPEC formula` (e.g., `SPEC AG (i=0 -> EX i!=0)`), and you can check it with the command `check_ctlspec`. You can find the syntax of CTL formulæ on the manual page.

1. Add to this file[1] the following properties in CTL. Then check them. (this code should not be dissimilar to last time).
   - There exists an execution in which the state is never busy.
   - In all executions, whenever a request is made, in all next reachable states, `state` is busy.
   - There is an execution in which whenever a request is made, in all next reachable states, `state` is busy.
   - In all executions, whenever a request is made, one of the next state has `state` busy.
   - There exists an execution in which the state is never busy without a previous request.

Exercise 2: (A microwave) This code[2] models the behaviour of a microwave.

1. Draw (on a paper) the automaton described by this model (and give relevant names to transition – after all this is a microwave, that should be doable).
2. Express in CTL and check the following properties:
   - whenever the heat is on, opening the door causes an error.
   - every error remains present until the start button is off.

[1] `./first-example-bis.smv`
[2] `./microwave.smv`

- it is always possible to eventually turn the heat on.
- there is an execution in which at every moment, every sequence of actions will eventually turn the heat on.
- before the heat is on, the start button must be pressed.
- if the machine is forever in the error state, then the door is opened or closed in each step.

Exercise 3: (Modelling an elevator)

The goal of this exercise is to model an elevator in smv, and to check its behaviour in CTL. Your elevator can navigate between floors 1 to N (with a fixed value of N which I let you choose – but don't take it too big). It has a set of buttons, 1 for each floor, which can be on or off. When a button is on, that means the elevator has been called to that floor (and should go there). At each step, the elevator can go up or down by one step, and can be called to some floors. When it is at a given floor, the corresponding button is switched off.

1. Make a model of such an elevator. The following properties will have to be expressed in CTL, included to your model, and true. You may need some more information in your model than what I've discribed to have a correct model.
   - If no button is on, the elevator does not move.
   - When the elevator is called to a floor, it will eventually reach it in every execution.
2. In addition, check on it the following properties (in CTL). Some may be false.
   - Whenever a button is on, the elevator does not stay put.
   - If the elevator is at a given floor, the corresponding button is off.
   - There is an execution in which all buttons are false at the same time infinitely often.
   - In all executions, all buttons are false at the same time infinitely often.
   - In all executions, all buttons are false infinitely often.
   - There is an execution in which there is always one button on (except at the start, be careful about that).

Exercise 4: (Mutual exclusion (le retour de la vengeance)) Even if processes are not recommended in smv (see next exercise), it is still possible to model a mutual exclusion algorithm in smv. Here[3] is a code of it.

1. Draw the described system.
2. Write and check in CTL the following properties:
   - The two processes are never in the critical section at the same time.
   - For every execution, whenever a process tries to access the critical section, it will eventually reach the critical section in every subsequent executions (one formula for each process).

[3] `./mutex-sync.smv`

3. Would it be easy to extend that modelisation to more than two processes?

Exercise 5: (Processes (deprecated)) If we have time, we will quickly discuss of a feature which should disappear in a next version (announced since some years now though): processes. In nuXmv, it is not possible to have cycling dependency as it was the case in Spin. With processes, it used to be the case. Here[4] is an example of that behaviour, which should not be unfamiliar. Understand it and check it, but do not put to much time in this: this feature is not supposed to be used.

[4] `./semaphore.smv`