Exam Program Verification 2008/2009
23-09-2009, 09:00-11:00

Lecturer: Wishnu Prasetya

1. [2 pt] Consider the following Promela model, consisting of 3 processes.

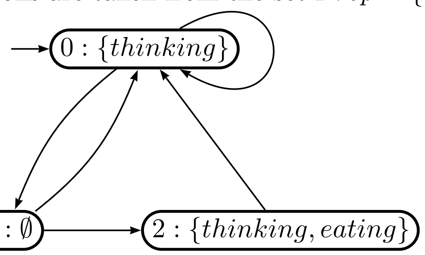
```
chan select = [0] of {bit} ;
chan x1 = [0] of {byte} ;
chan x2 = [0] of {byte} ;

active proctype SELECT () {
   bit b = 0 ;
   do
   :: atomic { b=!b; select!b } /* Alternatingly send 0 and 1 */
   od
}

active proctype STREAM() {
   do /* can send 0s on channel x1, and 9s on channel x2 */
   :: x1!0
   :: x2!9
   od
}

byte port ;

active proctype MUX(){
   bit b ;
   byte v1 ; byte v2 ;
   do
   :: select?b ; x1?v1 ; x2?v2 ;
      d_step {
        if /* decide the value on port based on value of b */
        :: b    -> port = v1 ;
        :: else -> port = v2 ;
        fi
      }
  od
}
```

   Express the following requirements in SPIN. You are free to use whatever verification approach supported by SPIN (option, assertion, LTL, etc).

   (a) The system does not dead-lock.

   (b) `MUX` will alternately put 0 and 9 in the varible `port`.

2. [2.4 pt] Consider the Kripke structure K given below. The states are numbered (0,1,2). Each state has been labelled by the set of atomic propositions that hold in the state. The propositions are taken from the set $Prop = \{thinking, eating\}$.



The questions:

(a) Let π be an (infinite) execution of K, and let i be a natural number. Give a formal definition of:

$$\pi, i \ \vdash \ \phi \ \mathsf{U} \ \psi$$

where ϕ, ψ are arbitrary LTL formulas.

(b) Give a Buchi automaton L that represents the LTL formula:

$$thinking \ \ \mathsf{U} \ \ \mathsf{X}(thinking \wedge eating)$$

(c) Construct the Buchi automaton $M = K \cap L$.

(d) So, does the following property hold? (note the negation)

$$K \ \ \vdash \ \ \neg(thinking \ \ \mathsf{U} \ \ \mathsf{X}(thinking \wedge eating))$$

If you think the property holds, explain why. Explain this in terms of M and the acceptance criterion of a Buchi automaton.

If you think the property does not hold, give an (infinite) sentence of M as your counter example. Explain why this sentence is a counter example in terms of M and the acceptance criterion of a Buchi automaton.

3. [2.4 pt] Consider again the Krikpe structure K from No. 2. We will encode each state by the following boolean functions:

State	Encoding
0	$\overline{x}\,\overline{y}$
1	$\overline{x}y$
2	$x\overline{y}$

where $\overline{f}$ denotes $\neg f$, and fg denotes $f \wedge g$.

(a) Give a boolean formula that encodes the automaton K.

(b) If ϕ is a CTL formula, let W_ϕ denotes the set of states of K on which ϕ holds. More precisely, W_ϕ consists of all states s of K such that $K, s \vdash \phi$.

Give a boolean formula that encodes $W_{\mathsf{EX}(thinking \wedge eating)}$.

(c) We will now calculate $Z = W_{\mathsf{E}(thinking \ \mathsf{U} \ \mathsf{EX}(thinking \wedge eating))}$, but we will do so symbolically (via boolean formulas). This is calculated iteratively.

Give boolean formulas that encode Z_0, Z_1, and Z_2.

(d) So, does K satisfies the specification:

$$\mathsf{E}(thinking \ \mathsf{U} \ \mathsf{EX}(thinking \wedge eating))$$

? Explain your answer.

4. [2.1 pt] Consider this CSP processes:

$$\begin{array}{rcl} P & = & (a \rightarrow STOP) \Box (a \rightarrow Q) \\ Q & = & (b \rightarrow STOP) \sqcap (a \rightarrow P) \end{array}$$

The alphabets of both P and Q are $\{a, b\}$.

(a) Give all failures of P whose traces are of length 1.

(b) Give a non-deterministic automaton M_P that generates exactly the same set of failues as P. You need to label each state of M_P with its refusals.

(c) Does the process $a \rightarrow b \rightarrow STOP$ refines P? Explain your answer.

5. [1.1 pt] We want to write a tactic `DROP` that drops its first assumption. So,

$$\texttt{DROP}\ (t{::}A\ ?\!-\ u) \quad = \quad A\ ?\!-\ u$$

where $t{::}A$ means t in front of the list A (what you in Haskell would write $t{:}A$).

In this exercise I want you to construct this tactic *explicitly*. A tactic is a function of this type:

$$\begin{array}{lcl} \texttt{type}\ tactic & = & goal \rightarrow (goal\ list\ *\ proofFunction) \\ \texttt{type}\ goal & = & term\ list\ *\ term \\ \texttt{type}\ proofFunction & = & thm\ list \rightarrow thm \end{array}$$

where $A * B$ denotes the type of pairs over A and B (what you in Haskell would write (A, B)). Here is a template to write `DROP`; you need to complete it:

```
fun DROP_TAC (t::A,u) =
   let
   fun proofFunction thms = ...
   val newgoals = ...
   in
   (newgoals,proofFunction)
   end ;
```

To help you, you are given the following inference rule $R : term \rightarrow thm \rightarrow thm$ that can weaken a theorem like this:

$$R\, t\, (A \vdash u) \quad = \quad A \vdash t \Rightarrow u$$