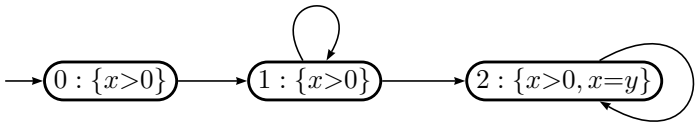
Exam Program Verification 2007/2008
UNNIK-211, 01-07-2008, 09:00-12:00

Lecturer: Wishnu Prasetya

June 30, 2008

1. [2.5 pt] Consider this Kripke structure K:



The set of states of this automaton is $\{0, 1, 2\}$, with 0 as the initial state. The set *Prop* of atomic observations is $\{x{>}0,\ x{=}y\}$. The labeling of every set with these observations are as given in the above picture.

We are going to do LTL model checking to verify whether the above 'program' satisfies the property: $\mathsf{true}\ \mathbf{U}\ (x{=}y)$.

(a) Draw a Buchi automaton B representing $\neg(\mathsf{true}\ \mathbf{U}\ (x{=}y))$.

(b) Give the formal definition of the Buchi automaton you draw in (a). That is, describe it in terms of a tuple (Σ, Q, ρ, I, F); what are your Σ in this case, your Q, your ρ and so on.

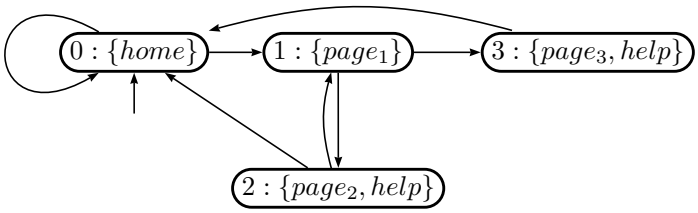
(c) Construct the automaton $K \cap B$. Explain why we need this automaton for model checking your property.

(d) So, according to your $K \cap B$, does your original program K satisfy the property $\mathsf{true}\ \mathbf{U}\ (x{=}y)$?

If yes, explain why. If no, give your counter example in terms of a path in $K \cap B$.

(e) SPIN uses a nested DFS algorithm to find a counter example. Explain why a single DFS is not sufficient.

2. [2.5 pt] Consider this Kripke structure K:



The set of states of this automaton is $\{0, 1, 2, 3\}$, with 0 as the initial state. The set *Prop* of atomic observations is:

$$\{home, page_1, page_2, page_3, help\}$$

The labeling of every set with these observations are as given in the above picture.

(a) We want to express that when *home* is true, there *exists* a path leading to a state where $page_3$ is true. How to express this in CTL?

(b) Can we express the same thing in LTL? If yes, give the formula. If no, motivate why not.

(c) We will do CTL model checking to check if the automaton K above satisfies the property $\mathbf{A}[\mathsf{true}\ \mathbf{U}\ help]$. The model checking algorithm proceeds by iteratively labeling the states of K with formulas. Fill in the following tabel to reflect the first 4 iterations of your model checking procedure:

Iteration	State			
	0	1	2	3
0	$\{home\}$	$\{page_1\}$	$\{page_2, help\}$	$\{page_3, help\}$
1	$\{home\}$	$\{page_1\}$	$\{page_2,\ help,\ \mathbf{A}[\mathsf{true}\ \mathbf{U}\ help]\}$	$\{page_3, help,\ \mathbf{A}[\mathsf{true}\ \mathbf{U}\ help]\}$
2				
3				

(d) When should we terminate the iterations in the above model checking procedure?

(e) Does K satisfy $\mathbf{A}[\mathsf{true}\ \mathbf{U}\ help]$, according to your model checking? Explain.

3. [2.5 pt] Consider the two CSP processes given below. The alphabet of both is $\{a, b\}$.

$$P_1 = (b \rightarrow STOP) \ \square \ (a \rightarrow a \rightarrow P_1)$$

$$P_2 = (b \rightarrow STOP) \ \sqcap \ (a \rightarrow a \rightarrow P_2)$$

(a) Describe the kind of traces that can be generated by P_1 and P_2. So, how are these processes ralated to each other in terms of trace-based refinement?

(b) Give all failures of P_1 respectively P_2 whose traces are of length 2 or less. You can use tables like the one below to list those failures.

E.g. for P_1:

trace length	failures
0	$(\langle\rangle, \emptyset), ...$
1	$(\langle a\rangle, \emptyset), (\langle a\rangle, \{b\}), (\langle b\rangle, \emptyset), (\langle b\rangle, \{a\}), (\langle b\rangle, \{b\}), (\langle b\rangle, \{a, b\})$
2	...

(c) Does $P_1 \sqsubseteq P_2$ hold under the failures semantic? How about $P_2 \sqsubseteq P_1$?

(d) Draw the automata M_1 and M_2 representing P_1 and respectively P_2. Label each state of these automata with its refusals.

(e) Construct the automaton $M_1 \cap M_2$, and explain how we can check $P_1 \sqsubseteq P_2$ using $M_1 \cap M_2$. So, does $P_1 \sqsubseteq P_2$ hold according to your refinement checking procedure?

4. [2.5 pt] Consider the following concepts:

> "Abstractly, a program P can be seen as a function that maps an initial state to a set of possible end-states. If P does not terminate when executed on an initial state s, we will express this by mapping s to an empty state. That is, in our abstract representation $P\ s = \emptyset$.
>
> A program P always terminates if for *all* (intial) state s, $P\ s$ is not empty. It follows that if P and Q are two programs that always terminate, so does $P; Q$."

We want to express those concepts in HOL (to eventually prove the claimed theorem, though we will not do so here).

(a) Give a HOL type that will be sufficient to represent the above abstract concept of "program", then give HOL definitions that capture the concepts "P always terminate" and "$P; Q$".

(b) Write a formula capturing the theorem "if P and Q are two programs that always terminate, so does $P; Q$"

(c) In HOL a goal has the following type:

$$\texttt{type}\ goal\ =\ (term\ list) * term$$

and a tactic has the following type:

$$\texttt{type}\ tactic\ =\ goal \rightarrow ((goal\ list)\ *\ (thm\ list \rightarrow thm))$$

So, when given a goal v, a tactic tac will produce a pair (z, f). Explain the roles of z and f and their relations to v.

(d) Let's now apply the above understanding. Write the combinator `TRY` that will behave as follows. Given a tactic tac, `TRY tac` will apply tac on the given goal. If it succeeds, then we are done. However if tac fails on the goal (that is, if it throws an exeception), then we do nothing with the original goal (and throws no exception).

You can write `TRY` in ML, Haskell, or even in some pseudo imperative language.