Exam-2 Program Verification 2017/2018

BBG-001, 8th Nov. 2017, 9:00 - 12:00

Lecturer: Wishnu Prasetya

1. **Formalizing Properties** [1.5 pt].

   Let $P, Q, ... \in \mathit{Process}$ and $r, s, ... \in \mathit{Resource}$ be processes in a system and resources that the processes may want to use. Formalize the following properties; you can use either LTL, CTL, or CTL*. You can introduce state predicates to abstractly capture needed concepts, e.g. $req(P, r)$ can be a state predicate modeling the fact that P is currently requesting the resource r, if the predicate is true, and otherwise it is not currently requesting r.

   (a) P and Q are not allowed to access r at the same time.

   (b) The system should be weakly fair. That is, if a process P repeatedly requesting a resource (it does not need to persistently maintain the request), P should eventually get access to the resource.

   (c) Whenever P is requesting access to r, if after maintaining this request for some time it still does *not get the access*, it should be possible for P to cancel this request.

   (d) At all times, there should be at least one resource that is not used by any process.

   (e) It should be possible for P to use s right after it has used r.

2. **Expressing LTL as an FSA** [1.5 pt].

   Give for each of the LTL formula below, a Buchi automaton that equivalently describes the formula. You can use either ordinary or generalized Buchi automata. For each automaton, please explicitly **specify its groups of accepting states**.

   (a) $\neg(p \;\mathbf{U}\; (p \wedge q))$

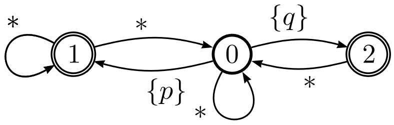
   (b) $p \;\mathbf{U}\; ((\neg p \wedge q) \;\mathbf{U}\; r)$

   (c) $\Diamond\Box\Diamond\Box p$

3. **LTL model checking** [3 pt].

(a) Consider the following generalized Buchi automaton K. The set of state predicates to consider is $Prop = \{p, q\}$. An arrow of the form $s \xrightarrow{*} t$ means that a transition from s to t is always possible with any label from 2^{Prop}.

K, 0 is the initial state:

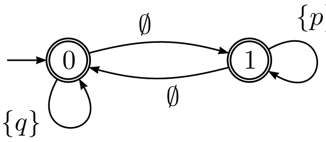


We have **two groups** of accepting states, namely: $F_1 = \{1\}$ and $F_2 = \{2\}$.

Give an equivalent **ordinary** (non-generalized) Buchi automaton that represents the same language. (Hint: which LTL property does the above automaton represent?)

(b) Below is an ordinary Buchi automaton, M, representing a program. The $Prop$ is the same as K above.

M, 0 is the initial state:



All states are acceptance states.

i. Construct the automaton that represents $M \cap K$.

ii. Let ϕ be the LTL formula represented by K. Is $\neg\phi$ a valid property of M? Explain your answer in terms of $M \cap K$.

4. **Model checking of programs with concrete states** [1 pt].

(a) In Promela, a system is made of a finite number of concurrent processes. Each process is sequential, and can be thought to maintain a program counter, whose value points to the next statement in the process to execute.

How can you use LTL model checking to check if a Promela system contains a 'dead' statement? A dead statement is a statement that will never be executed.

(b) Below we describe a system consisting of 3 processes (in Promela notation), namely P, Q, Obs. The system operates on global variables x, y and a global channel c; x, y are of type integers and are initially 0, and c is a channel of size 0.

The process P keeps increasing the value of x, modulo 3. Q waits until x>0, and then sends the value of x over the channel c. Note that (x>0) in the code of Q is a blocking expression a la Promela.

active proctype P {**do** :: $x = (x+1)$ **mod** 3 **od**}
active proctype Q {(x>0) ; $c!x$}
active proctype Obs {$c?y$}

What are the possible values that Obs can receive from the channel c?

(c) Construct a concrete state finite state automaton representing $P||Q||Obs$.

5. **CTL model checking** [1 pt].

   Consider again the program represented by the Buchi automaton M in the question 3b. Consider the CTL property:

   $$\phi \ = \ \mathbf{AF}(\mathbf{E}[p \ \mathbf{U} \ (\neg p \wedge \neg q)])$$

   Give an algorithm to model check ϕ on M. (Note that M is not a Kripke structure)

6. **Symbolic model checking** [1 pt].

   Consider a Kripke structure $M = (S, s_0, R, Prop, V)$ with $Prop = \{p, q\}$. S consists of 8 states. The transition relation R is symbolically represented by the Boolean formula below:

   $$(\bar{x}.y') \ \vee \ (\bar{y}.z') \ \vee \ (\bar{z}.x')$$

   We write for example $e_1.e_2$ as a shorthand for $e_1 \wedge e_2$. We write $\bar{e}$ to mean the negation $\neg e$.

   Above, x, y, z are boolean variables representing the states in S. In the transition relation above, the primed version of these variable, x', y', z' represents the next-states of the transitions that the relation describes.

   We assume the initial state $s_0 \in S$ to be represented by the formula $\bar{x}.\bar{y}.\bar{z}$.

   In M, the function V describes the labeling of p and q on the states in S. This labelling is now encoded with the following Boolean formulas, for p and q respectively:

   $$\begin{array}{lcl} W_p & = & x.z \vee y \\ W_q & = & \bar{y} \end{array}$$

   Questions:

   (a) Give a Boolean formula that represents the set of all successor states of the initial state s_0.

   (b) How to check if $p \to q$ is a valid property of M?

   (c) Is **EX** q a valid property of M? Explain.

7. **BDD-based model checking** [0.5 pt].

   (a) Explain how to use BDD to check if a given state t is reachable from the initial state of a program with finite number of states.

   (b) Represent the program M from the question No. 6 as an ordered and reduced BDD. Mention your ordering.

8. **Model checking** [0.5 pt].

   Let K and M be two **generalized** Buchi automata. They have indeed finite number of states. Their labels are taken from **the same** finite set of alphabets. We write $K \sqsupseteq M$ (K 'refines' M) to mean that all sentences accepted by M are also accepted by K. Propose a way to check this in finite number of steps.