Exam-2 Program Verification 2016/2017

BBG-209, 30th Jan. 2017, 13:30–16:30

Lecturer: Wishnu Prasetya

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

   Let P be a process, and r a resource that P may want to use. Formalize the following properties in LTL or CTL. You can introduce state predicates to abstractly capture needed concepts, e.g. $req(P, q)$ 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) The description of P mentions that that are some states that can be called 'idle states'. If P is in such state, we say that P is idle, and otherwise non-idle. Express this property: as long as P remains idle, it will not request access to the resource r.

   (b) If P is idle, it should be *possible* for P to become non-idle. Note that we do not want to require it to eventually become non-idle. We do want to insist that there should be a possibility to become non-idle.

   (c) Whenever P requests to access the resource r, it will maintain the request, and eventually the request should be granted.

   (d) When P is granted access to the resource r, it should be *possible* for P to later release r *without* in the mean time ever uses r.

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

   Give for each of the LTL formula below, a (generalized) Buchi automaton that equivalently describes the formula. For each automaton, specify which states are the initial states, and what are its groups of accepting states.

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

   (b) $\Box(p \ \mathbf{U} \ q)$

   (c) $(\Box\Diamond p) \ \wedge \ (\Box\Diamond\mathbf{X} \ q)$

   (d) $\Diamond\Box\Diamond\Box p$

3. **Concepts related to FSA** [1 pt].

   Let $M_1 = (S_1, i_1, F_1, R_1, \Sigma_1)$ be a Buchi automaton, where S_1 is its set of states, i_1 is its (single) initial state, F_1 is its set of accepting states, Σ_1 is the alphabet of the labels on M_1's transitions, and R_1 describes the transitions, such that if s is a state and $a \in \Sigma_1$, then $R_1(s, a)$ describes transitions from s labelled by a. There is a transition (arrow) labelled with a that goes from the state s to a state t if and only if $t \in R(s, a)$.

   Analogously, let $M_2 = (S_2, i_2, F_2, R_2, \Sigma_2)$ be another Buchi automaton.

   Give an algorithm to construct a Buchi automaton M_3, such that $L(M_3) = L(M_1) \cap L(M_2)$.

4. **LTL model checking** [1.5 pt].

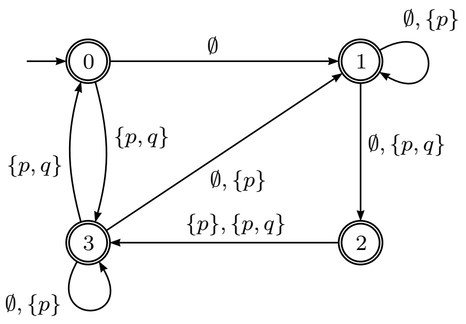
   Consider a program M modeled by a Buchi automaton shown below (left). 0 is the initial state. All states are accepting. The set of state properties to consider is $Prop = \{p, q\}$. The arrows are explicit labelled by subsets of $Prop$.

   If a transition from s to t has multiple labels, it means that we can go from s to t, through *any* of the labels.

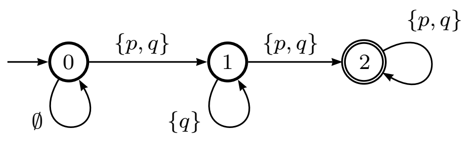
   Consider an LTL property ϕ that we want to verify on the program. The Buchi automaton K below represents the formula $\neg\phi$. The initial state is 0, and only state 2 is accepting.

   Both Buchi automata are ordinary Buchi automata (not generalized ones).

   M :



   K :



   (a) What is this ϕ? (express it in LTL; you can use $\neg$)

   (b) Construct $M \cap K$. Is ϕ valid? Explain your answer in terms of $M \cap K$.

5. **CTL model checking** [1.5 pt].

   Let a program M be modeled by a Kripke structure $(S, i_0, R, Prop, V)$, where S is M's set of states, i_0 is its single initial state, R is a function describing the transitions between the states, $Prop = \{p, q\}$ is a set of state properties (predicates), and V is a function that labels the states with members of $Prop$.

   Consider a CTL property $\phi = \mathsf{E}(p \ \mathsf{U} \ \mathsf{A}(q \ \mathsf{U} \ (p \wedge q)))$. Give an algorithm to model check ϕ on M.

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

   Consider a Kripke structure $M = (S, i_0, R, Prop, V)$ with $Prop = \{p, q\}$. S consists of 8 states. The transition relation R is symbolically represented by the Boolean formula below:

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

   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 i_0 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:

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

Questions:

(a) Is p a valid property of M? Explain.

(b) Is $\mathsf{AX}\ q$ a valid property of M? Explain.

7. **CSP** [1 pt].

Consider the following CSP processes P_1 and P_2. The alphabet of both is $\{a, b\}$. The $\to$ operator is right associative; so $a \to b \to P$ means $a \to (a \to P)$.

$$\begin{aligned} P_1 &= (a \to \mathsf{STOP}_{\{a,b\}}) \,\Box\, (b \to a \to P_1) \\ P_2 &= b \to a \to a \to (\mathsf{STOP}_{\{a,b\}} \sqcap (a \to \mathsf{STOP}_{\{a,b\}})) \end{aligned}$$

(a) Does $P_1 \sqsubseteq P_2$ hold under the trace semantic? Explain.

(b) Consider another process Q with alphabet $\{b, c\}$ defined by:

$$Q = (b \to c \to Q) \sqcap \mathsf{STOP}_{\{b,c\}}$$

Give an Finite State Automaton (FSA) that equivalently describes Q under the failures semantic. Label each state in your FSA with the *initials* and *refusals* of that state.

(c) Give $\mathsf{refusals}(P_1 || Q)$.

8. **LTL model checking** [0.5 pt].

Suppose we allow a Promela model P to have infinite number of states. Indeed, we cannot verify such a model through model checking. However, we can still do testing. Suppose we want to have a set of test cases with 100% actions-coverage. That is, every atomic action in P should have been executed at least once during the executions of all these test cases. Note that the test cases only need to fulfill these requirement collectively, rather than individually.

Propose an idea of how we can exploit LTL model checking to automatically generate such a set of test cases.

9. **CTL model checking** [0.5 pt].

In the standard CTL, the atomic formulas are state predicates. Suppose we extend CTL by allowing LTL formulas to appear as CTL's atomic formulas. The meaning is formally defined as follows. Let $M = (S, i_0, R, Prop, V)$ be a Kripke structure. For simplicity it has a single initial state i_0. Let t be a computation tree of M, rooted in some state in S. The notation $\mathsf{root}(t)$ denotes this state. Let f be an LTL property. Its meaning in our CTL extension is as follows:

$$M, t \models f \ \overset{def}{=}\ M_{\mathsf{root}(t)} \models_{LTL} f$$

where $M_u \models_{LTL} f$ means that f is valid in LTL, with respect to the Kripke structure M where we replace its initial state with u.

Propose a modification to CTL model checking to model check formulas of the above described extended-CTL.