Exam Software Verification (TDDE34)

2024-06-01

Jour: Ahmed Rezine (1938).

- You may answer in either English or Swedish.
- Write clearly. **Unreadable text will be ignored.**
- Be precise in your statements. Ambiguous formulations will lead to reduction of points.
- Motivate clearly all statements and reasoning.
- The exam is 40 points and graded U, 3, 4, 5 (**preliminary** limits: 20p, 30p, 35p). Points are given for motivations, explanations, and reasoning.

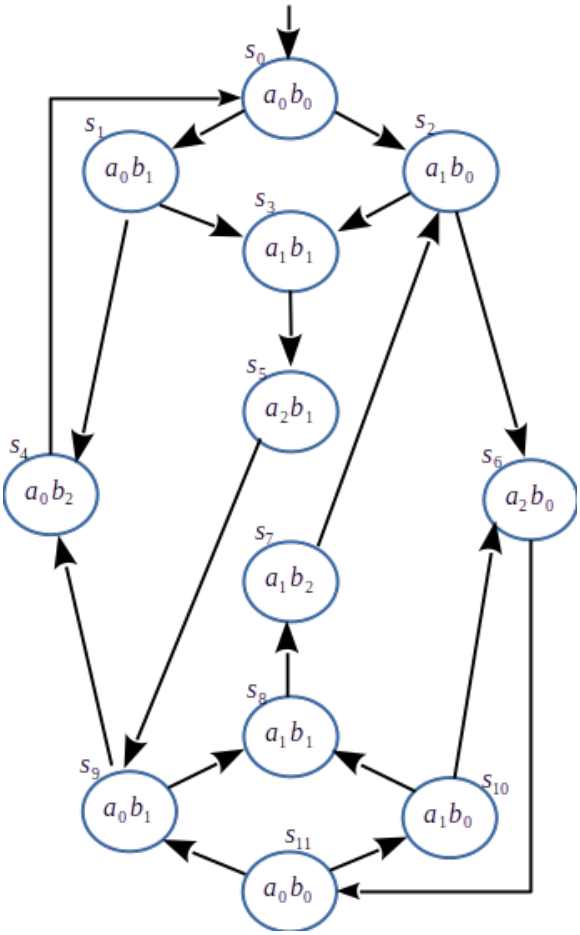


Figure 1: Kripke structure where s_0 is the initial state.

1 Branching time (10p)

Figure 1 depicts a Kripke structure with 12 states $\{s_0, s_1, \ldots, s_{11}\}$ where s_0 is the initial state. Each state is annotated with the atomic predicates that hold in it. For instance, only a_2 and b_1 hold in s_5. A CTL state formula holds in a number of states. For instance:

- The CTL state formula a_0 only holds in $\{s_0, s_1, s_4, s_9, s_{11}\}$.
- The CTL state formula $(a_1 \wedge b_1)$ (i.e., the conjunction of a_1 and b_1) only holds in $\{s_3, s_8\}$.

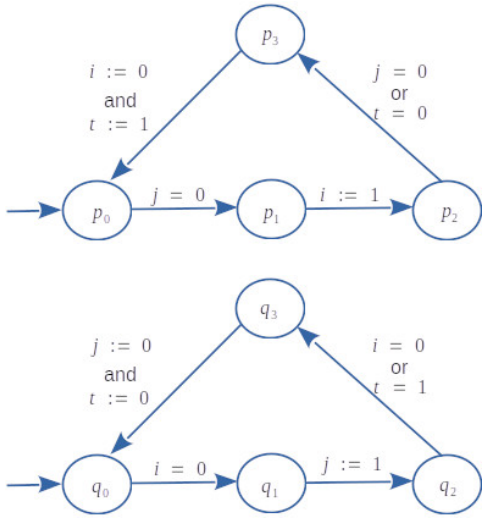
For each state formula in the following, give the exact set of states where the formula evaluates to true. Missing a state where the formula is true, or adding

a state where it is false, results in a wrong answer. Recall **X** is used to mean "next state on a path" and the exclamation mark ! is used for negation:

1. $\mathbf{AX}(b_2)$ (2p)
2. $\mathbf{EX}(b_2)$ (2p)
3. $\mathbf{EG}(!b_2)$ (3p)
4. $\mathbf{AG}\,(b_1 \Rightarrow \mathbf{AF}(b_2))$ (3p)

2 Mutual exclusion (10p)

Assume the following description of a simple mutual exclusion algorithm for two processes `p` and `q`. State p_0 (resp. q_0) is the initial state of process `p` (resp. process `q`). Process `p` (resp. process `q`) moves to state p_1 (resp. state q_1) in case it wants to access its critical section. State p_3 (resp. q_3) is the critical section of process `p` (resp. process `q`). Variables `i`, `j` and `t` are shared by the two processes and can only be tested or assigned to. Initially, all of them are 0. Transitions can test variables' values. For instance, process `p` can move from state p_0 to p_1 (respectively from state p_2 to p_3) if variable `j` is 0 (respectively variable j is 0 or variable t is 0). Transitions can also assign values to variables. For instance, process `p` can move from state p_1 to p_2 (respectively from state p_3 to p_0) while assigning value 1 to `i` (respectively value 0 to `i` and value 1 to `t`). Transitions are atomic and we assume interleaving semantics. This means one process moves at a time without interruption by the other process during each single transition. Possible consecutive sequences of transitions include process `p` moving from p_0 to p_1 then process `q` moving from q_0 to q_2 via q_1, then process `p` moving from p_1 to p_0 via p_2 and p_3.



In the following, we use `@pn` (for $n : 0 \leq n \leq 3$) to mean the proposition stating process `p` is at state p_n. We do the same for process `q`. For instance, the proposition `@q3` is true when process `q` is at its critical section.

Answer the following questions:

- Give a Büchi automaton for the LTL formula capturing all infinite runs that respect mutual exclusion. Your Büchi automaton may use true, false, and Boolean combinations of `@pn` and `@qm` (for $n, m : 0 \leq n, m \leq 3$). (2p)
- Is there a run that violates mutual exclusion? If yes, give such a run, otherwise clearly argue why there are no runs that violate mutual exclusion (3p)
- The LTL formula φ defined as $G(\texttt{@p1} \implies F(\texttt{@p3}))$ captures that `p` does not starve. Give a Büchi automaton for the LTL formula φ. Your Büchi automaton may use true, false, and Boolean combinations of `@pn` and `@qm` (for $n, m : 0 \leq n, m \leq 3$). (2p)
- Suppose a scheduler ensures `p` is chosen infinitely often (i.e., it cannot start forever ignoring `p` after a finite amount of transitions). Is there a run that violates φ? If yes, give such a run, otherwise clearly argue why φ holds. (3p)

3 Symbolic representation (8p)

1. Assume integer variables x_1 and x_2. The following formula involves expressions in Linear **Integer** Arithmetic (LIA). Assume $a \wedge b$ stands for conjunction of a and b, and $p\%2$ is 0 if p is a multiple of 2 and 1 otherwise.

$$\left(\begin{array}{c} (x_1 + x_2 \leq 3) \wedge 0 \leq x_1 \wedge 0 \leq x_2 \\ \wedge \\ x_2 = x_1 + 1 \wedge (x_2 \% 2 = 1 \Rightarrow x_1 = x_2) \end{array} \right)$$

   Give a model (i.e., values for the variables) for the formula if it is satisfiable, otherwise argue why it is not satisfiable. (4p).

2. Let $f(v_0, v_1, v_2)$ be the formula $(v_0 \texttt{ xor } v_1) = v_2$ where v_0, v_1 and v_2 are Boolean variables, `xor` stands for the "exclusive or" binary operator (i.e., a `xor` b is 1 iff a and b are different). Give a BDD for f assuming the order $v_0 < v_1 < v_2$ (i.e., starting from the root, variable v_0 appears first, then variable v_1, ... etc). (4p).

4 Partial and total correctness (12p)

Consider the following simple program (all variables are natural numbers):

$$
\begin{array}{l}
\{Q : 0 \leq n \wedge i = 0 \wedge y = 1\} \\
\ \textbf{while}(i < n)\{ \\
\quad y := 2 * y; \\
\quad i := i + 1; \\
\ \} \\
\{R : f(y, n)\}
\end{array}
$$

- Give an expression for $f(y, n)$ (e.g of an expression could be $y = 3 \times n$). The expression you propose should result in the most precise (i.e., strongest) post-condition R that always holds at the end of the loop when starting from Q and that relates the values of y and n (and only the values of y and n, possibly with some constants). If you find it helpful, you can use symbols such as $\Sigma_{k=a}^{b} g(k)$ (respectively $\Pi_{k=a}^{b} g(k)$) for the sum (respectively, product) of all elements $g(a), g(a+1), ...g(b)$. Observe the condition $R : y = 3 \times n$ does not satisfy these requirements. (2p).

- Find a suitable invariant and use it to show that if the loop terminates after starting from a state satisfying Q then it terminates in a state satisfying R. (6p).

- Find a suitable variant function and use it to show the loop terminates. (4p).