EECS 219C: Computer-Aided Verification **Sanjit A. Seshia**

Homework 2: Temporal Logic and Explicit-State Model Checking

Assigned: February 14, 2007 *Due in class: February 28, 2007*

Note: See the webpage for rules on collaboration.

1. **Expressing Properties in LTL** (20 points)
   Let p, q, r denote atomic propositions.

   Express the following properties in LTL: (5 points each)

   (a) The transition from $\neg p$ to p will happen at most once.

   (b) There are infinitely many transitions from $\neg p$ to p (and vice versa).

   (c) The lights of a traffic signal light in the following periodic sequence: red, yellow, green, red, yellow, green, ... Only one light is on at any given time, starting with red, but you cannot make any assumptions on how long a light is on other than it will be on for a finite length of time.

   Use r, y, g as atomic propositions denoting the states of the traffic light (light = red, light = yellow, light = green).

   (d) If q holds in a state s_i and r holds in some subsequent state (s_j, $j > i$) then p must not hold in any state s_k in between ($i < k < j$, where j is the first such subsequent state).

2. **LTL, CTL, CTL*** (25 points)

   (a) (5 points)
   Slide 26 of Lecture 6 states that the CTL formula **AF AG**p is stronger than the LTL formula **FG**p and gives an example state machine satisfying the latter but not the former. (The start state of this example state machine is the left-most blue state.)

   Explain why that state machine does not satisfy the CTL formula. Also prove that if a system satisfies the CTL formula, it must also satisfy the LTL formula.

   (b) (10 points)
   Explain why the CTL* formula **E**(**GF**p) is not equivalent to either of the CTL formulas **EG**(**EF**p) and **EG**(**AF**p). For each pair of (CTL*, CTL), state which property is stronger, and exhibit a state machine that satisfies the weaker property but not the stronger property.

(c) (10 points)
Which of the following pairs of CTL formulas are equivalent? For those that are not, exhibit a model of one of the pair which is not a model of the other (with brief justification).

i. $(\mathbf{EF}p) \vee (\mathbf{EF}q)$ and $\mathbf{EF}(p \vee q)$.

ii. $(\mathbf{AF}p) \vee (\mathbf{AF}q)$ and $\mathbf{AF}(p \vee q)$.

3. **Stuttering equivalence and LTL** (15 points)
Two runs $\pi_s = \langle s_0, s_1, s_2, \ldots \rangle$ and $\pi_t = \langle t_0, t_1, t_2, \ldots \rangle$ are *stuttering equivalent* (denoted $\pi_s \sim_{st} \pi_t$) if there are two sequences of non-negative integers $0 = i_0 < i_1 < i_2 < \ldots$ and $0 = j_0 < j_1 < j_2 < \ldots$ such that for every $k \geq 0$,

$$L(s_{i_k}) = L(s_{i_k+1}) = \ldots = L(s_{i_{k+1}-1}) = L(t_{j_k}) = L(t_{j_k+1}) = \ldots = L(t_{j_{k+1}-1})$$

where L is the labeling function that maps a state to the set of atomic propositions true in that state.

In words, π_s and π_t are stuttering equivalent if they are equivalent (the same run) after collapsing finite sub-sequences of identically labeled states.

Note that the length of the corresponding finite sub-sequences in the two runs can be different; i.e., it could be that $i_k - i_{k-1} \neq j_k - j_{k-1}$.

Prove that, for any LTL formula ϕ that does not contain the "next" ($\mathbf{X}$) operator, if $\pi_s \sim_{st} \pi_t$, π_s satisfies ϕ if and only if π_t satisfies ϕ.

(Hint: Use induction on the size of the LTL formula, considering as cases the different kinds of LTL formula ϕ could be at the top level. Use formal definitions of a sufficient subset of LTL operators (see MC pg 29): e.g., since $\mathbf{F}$ is expressible using $\mathbf{U}$ you do not need to consider both.)

Note: The notion of stuttering is important for asynchronous systems where an arbitrary number of local transitions (those that do not change global state) may happen between global transitions. Partial order reduction creates machines whose runs are stuttering equivalent to those of the original machine. So, to safely use P-O reduction, the LTL formula should not mention the $\mathbf{X}$ operator. This is OK, since the $\mathbf{X}$ operator does not make very much sense in an asynchronous setting, since there is no global clock and what happens "next" may vary depending on the scheduling strategy.

4. **SPINing Elevators**[1] (40 points)

A SPIN model for an elevator controller with its environment is given at `http://www.eecs.berkeley.edu/~sseshia/219c/homeworks/elevator.pml`. The model describes the following working of the elevator:

- The elevator serves 4 floors, numbered 0 to 3, and up to 2 users.

[1] Problem originally formulated by Gerard Holzmann.

- Each floor has a button that a user can press to request the elevator to come to that floor and open the door.
- The elevator has 4 destination buttons, one for each floor, for the passenger to request transportation to one of the floors.
- If there are no requests, the elevator stays put with its doors open.
- There is one process for the controller, the elevator, and each of the users. The processes communicate by asynchronous message passing.
- A button push is a single event: holding the button down or releasing it has no further effect. Button pushes are modeled as message sends to the controller process.
- Status information is kept in global variables `floor`, `doors_open`, and `iam[..]`. These can be read by any process, but only one designated process writes to each of them.
- The controller schedules all actions of the elevator.

For this problem, you should first familiarize yourself with the Promela syntax and usage of the SPIN system. It is easiest to get started using the GUI-based version of SPIN called XSPIN (it is recommended to first read `http://spinroot.com/spin/Man/GettingStarted.html`). In "advanced options" under "set verification parameters", be sure to increase the search depth to 1,000,000. You might also need to increase the max memory usage bound.

Check the properties listed below by specifying them using either the "assert" feature or as an LTL property (note that LTL properties have to be compiled into Promela code for "never claims" – a Büchi automaton for the negation). For each property, state whether the model satisfies it or not. If not, study the error trace (the output of the "guided simulation" in XSPIN) and briefly explain the root cause of the error.

You are allowed to instrument the Promela code with "monitor variables" that simply observe and record changes in program counter and state, but do not affect the dynamics of the system. Any such change must be carefully justified and documented.

The properties are as follows: (in each case, state the property in LTL first; 10 points each)

(a) The elevator never moves with its doors open.

(b) Whenever a user at floor 1 requests the elevator and it is not at floor 1, then it eventually arrives at that floor and opens its doors.

(c) The elevator visits every floor infinitely often.

Print out a copy of the Promela specification with any changes/additions you made to it to verify the properties. Errors can be indicated on this printout.

If the number of users is increased by 1, will it change the results of your verification? How about a decrease by 1? (10 points)