

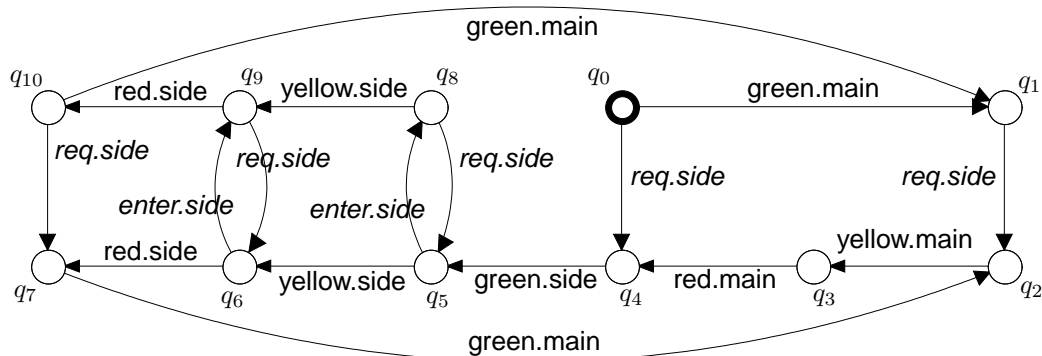
COMP 424-06A

Topics in Software Engineering

Assignment 2

Exercise 1 (5 marks)

Consider the following automaton, representing a misguided attempt to control the traffic lights from your assignment 1. This automaton can be considered as a Kripke-structure using the propositions and truth assignments given in the table.



Proposition	Meaning	States where true
g_m	Main street lights green	q_1, q_2
y_m	Main street lights yellow	q_3
r_m	Main street lights red	q_0, q_4, \dots, q_{10}
g_s	Side street lights green	q_5, q_8
y_s	Side street lights yellow	q_6, q_9
r_s	Side street lights red	$q_0, \dots, q_4, q_7, q_{10}$
req	Car waiting in side street	q_2, \dots, q_7

For each of the following CTL formulas, list all states where the formula is true.

- $\mathbf{AG} \neg(req \Rightarrow g_s)$
- $\mathbf{EG}(y_m \vee r_m)$
- $\mathbf{A}(g_m \mathbf{U} req)$
- $y_s \Rightarrow \mathbf{AF} r_s$
- $\mathbf{AG}(req \Rightarrow \mathbf{AF} g_s)$

Exercise 2 (15 marks)

An elevator system is to drive a *cabin* which can service four floors, numbered 0 to 3. The cabin can move up and down between the floors. At each floor, there is a position detector reporting when the cabin has safely reached the corresponding floor. The *cabin door* can be opened and closed. There are sensors to report if the door is completely open or completely closed. Finally, there are *request buttons* at each floor and in the cabin, by which users can request the elevator to travel to the different floors.



Signal	Meaning
<i>Sensors:</i>	
at[<i>i</i>]	The cabin is at floor <i>i</i> .
req[<i>i</i>]	There is a pending request to visit floor <i>i</i> .
is_open	The elevator door is completely open.
is_closed	The elevator door is completely closed.
<i>Actuators:</i>	
do_open	Open the elevator door.
do_close	Close the elevator door.
do_up	Move the cabin upwards.
do_down	Move the cabin downwards.

Download the elevator specification `exercise2_2.smv` from the course homepage

<http://www.cs.waikato.ac.nz/Teaching/COMP424A/>.

First read and understand the model. Then add appropriate properties to it and check them. If you find any bugs in the controller, fix them, and check the model again, until you are convinced that it behaves as you would expect of a sound elevator controller. Please do not change the plant model or the way how the plant and controller interact. Only make changes as indicated in the file.

In the end, you should be able to verify at least

- that the elevator door is safely closed when the cabin is travelling between floors;
- that the elevator never passes a floor with a pending request without servicing that request;
- that every request is eventually serviced.

In addition to producing a correct controller, please write a report of 1–2 pages about your experience with model checking on this example. Your report should discuss all the bugs you found in the controller from the course web site. It should include at least two of the properties that first failed, plus explanations of the corresponding counterexamples and how this has helped you to find and fix a bug in the controller. It should also explain your corrected controller and all the properties that you have checked, unless they are explained using comments in the `.smv` file.

Submission

Please hand in your solutions into the box marked COMP424A in front of room G 1.15. Your complete submission should include

- a) your written answers to exercise 1;
- b) a printout of your `.smv` file for exercise 2;
- c) a printout of the output obtained when running NuSMV on this file;
- d) your written report as specified in exercise 2.

In addition, please submit your `.smv` file for exercise 2 electronically through the assignment submission system at <http://byerley.cs.waikato.ac.nz/~tonym/submit/>. Please note that this system works only from our computing laboratories on campus.

Due date: Monday, 27th March 2006, 9:00 A.M.