Formal Methods in Software Engineering

Assignment 2 (Spin and Model-Checking)

(Due on Thu 15th Feb 2024. Total marks 80)

1. Consider the pairs of LTL assertions below. For each pair either give an informal proof that they are equivalent, or a counter-example model (an infinite sequence of propositional valuations) that shows they are not equivalent. Your counter-example should be in the form of an ulitmately periodic word of the form $u \cdot (v^\omega)$, where u and v are finite sequences of propsitional valuations over the propositions $\{p, q, r\}$. (*Marks 10*)

   (a) $G(Fp)$ and $F(Gp)$.

   (b) $\neg(pUq)$ and $(\neg q)U(\neg p)$.

2. Consider the Promela model below. (*Marks 20*)

```
byte x = 1;

proctype counter() {
  do
  :: x <= 1 -> x = x + 1;
  :: x >= 1 -> x = x - 1;
  od
}

init {
  run counter();
}
```

   (a) Describe the transition system that Spin compiles this model into. You may choose to ignore the intermediate states Spin uses to move to a satisfied guard. Show only the reachable states.

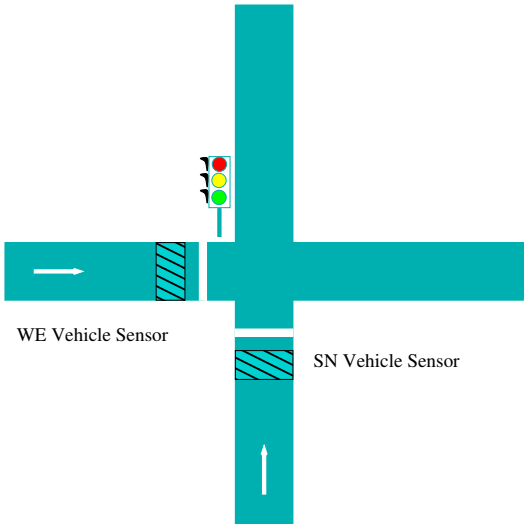
   (b) Does this model satisfy the property "$(x = 1) \implies F(x = 2)$"? Justify your answer.

   (c) Construct a state-based Büchi automaton (by hand if you like) for the *negation* of this formula, over the propositions p representing $x = 1$, and q representing $x = 2$.

   (d) Construct the product of the transition system of the model and the Büchi automaton for the formula above. Describe the counterexample path if any.

3. Construct the formula automaton for the LTL formula $true\,Up$, using the algorithmic construction described in class. You could treat *true* as $p \vee \neg p$. (*Marks 10*)

4. Imagine that you have been asked to implement a traffic light for a junction as shown in the figure below. The aim of this exercise is to use Spin to (a) design your system and model it in Promela, and (b) debug and eventually verify it, before you move on to implementing/fabricating it. (*Marks 40*)



Traffic at the junction moves in two directions: West-to-East (WE) or South-to-North (SN). There are two sets of lights: one for WE direction and the other for SN direction. Each light goes from green to amber for 1 sec and then to red; and from red directly to green. There are two sensors "WEsen" and "SNsen" which give a "true" signal whenever there is a vehicle waiting to cross in the respective directions. Your light subsystem runs at a certain clock speed, while the timer subsystem which counts seconds, sends clock ticks at a slower, non-deterministic speed. Your system should satisfy the following requirements:

(a) (Safety) The WE and SN lights are never green at the same time.

(b) (Utility) If a light is green and there is no vehicle waiting in the other direction, then the light should remain green.

(c) (Stability) The lights stay green for at least 3 sec at a stretch.

(d) (Liveness) Vehicles should not have to wait for more than 10 sec at a signal.

Build a Promela model of your design, taking into account these requirements, with three active proctypes, "tlight", "timer", and "vehicle" as shown below. Specify the last four requirements above as LTL properties in your Promela model.

```
mtype = { GREEN, AMBER, RED };
mtype = { GO, CHANGE, STOP };

bool tick = false;
bool WEsen = false;
bool SNsen = false;
WElight = GREEN;
SNlight = RED;
```

```
...

active proctype tlight() {
  bool ctr = 0;
  do
    ...
  od;
}

active proctype timer() {
  do
  :: tick = false;
  :: tick = true;
  od;
}

active proctype vehicle() {
  do
  :: SNsen = false;
  :: SNsen = true;
  :: WEsen = false;
  :: WEsen = true;
  od;
}
```

You could generate multiple versions of your design. For each version first test it out (manually or using simulate option of Spin) and convince yourself that your model is correct, before using the verify option in Spin. Your answer should contain a series of Promela models named `tlight-v1.pml`, `tlight-v2.pml`, ..., accompanied with a description of the issues you found (if any) while using Spin to verify the four properties, and how you plan to fix it in the next model. Submit as a zipped file `tlight-your-name.zip`.