Assignment 6 – LTL Monitoring 2

Exercise 1 (3 points) (Fairness in nuSMV)

The following example is from the nuSMV manual. The program represents a ring of three asynchronous inverting gates.

```plaintext
MODULE inverter(input)
VAR
  output : boolean;
ASSIGN
  init(output) := 0;
  next(output) := !input;

MODULE main
VAR
  gate1 : process inverter(gate3.output);
  gate2 : process inverter(gate1.output);
  gate3 : process inverter(gate2.output);
```

Among all the modules instantiated with the process keyword, one is nondeterministically chosen, and the assignment statements declared in that process are executed. Because the choice of the next process to execute is non-deterministic, this program models the ring of inverters independently of the speed of the gates.

Which of the following properties hold? Verify your answers with nuSMV.

- (a) gate 1 is infinitely often high
- (b) gate 1 always alternates between high and low
- (c) gate 1 can always eventually be high
- (d) gate 1 can always be low
- (e) in a fair execution, gate 1 cannot be always low
- (f) in a fair execution, gate 1 can be always low
- (g) in a fair execution, gate 1 is infinitely often high

Exercise 2 (6 points) (LTL Monitoring - application)

Consider your solution of the two-robot system from Assignment 2, Exercise 3. Assume that the pieces on the conveyor belts which are to be assembled can be of red, blue, or yellow color. Red pieces need to be treated with tool T1, blue pieces need tool T2, and yellow pieces can be treated with either T1 or T2.

- (a) Construct an example execution sequence of at least ten pieces on each of the two input conveyor belts. Simulate your model and note down the sequence of actions of the robots.
- (b) Write LTL formulas stating that every input piece is eventually on the output belt, after being treated with the tool fitting its color. Check these formulas with your monitor from Assignment 5, Exercise 3.
**Exercise 3** (8 points) (LTL Monitoring - extension)

Extend your monitor from Assignment 5, Exercise 3 such that it can deal with past-time operators. Recall that the semantics of $(φ U ψ)$ is defined by

$$(M, i) \models (φ U ψ) \text{ iff for some } j < i, (M, j) \models ψ \text{ and for all } j < k < i, (M, k) \models ϕ$$

That is, $(φ U ψ)$ cannot hold in the initial point of a finite (or infinite) sequence.

You get 3 extra points if your program works in linear time with respect to $M$, traversing the sequence only once. This is possible!

Would this extended monitor simplify your task from Exercise 1 (b) above?