CS 476 Homework #13 Due 10:45am on 5/5

Note: Answers to the exercises listed below should be emailed to `abir2@illinois.edu`, including the Maude code for the second exercise.

1. Answer the following questions:

   (a) Let $AP = \{p, q\}$. Explain — by giving in each case a counterexample trace $\tau \in [\mathbb{N} \to \mathcal{P}(AP)]$ such that, for ϕ the chosen wrong formalization, either $\tau \models p \rightsquigarrow q$ and $\tau \not\models \phi$ or $\tau \not\models p \rightsquigarrow q$ and $\tau \models \phi$ — why each of the following LTL formalizations ϕ of the "leads to" property $p \rightsquigarrow q$ is wrong:

   - $p \to q$
   - $\Box(p \to q)$

   (b) Write an LTL formula stating that for any path from the given initial state at only a finite number (possibly zero) of positions (i.e., steps) in such a path the property φ holds.

   (c) Write an LTL formula stating that for any path from the given initial state whenever φ holds at some step n, then it will also hold at all steps $n + 2k$ for any $k \in \mathbb{N}$.

   (d) Write an LTL formula stating that for any path from the given initial state, whenever φ holds at some step n, property ϕ will then hold at a strictly later step after that.

   (e) Write an LTL formula stating that for any path from the given initial state, whenever φ holds at some step n then ϕ does not hold at step n but will hold at a strictly later step after that.

   (f) Let $AP = \{p, q\}$. Define traces $\tau_1, \tau_2 \in [\mathbb{N} \to \mathcal{P}(AP)]$ such that, instantiating φ to p, and ϕ to q, τ_1 satisfies property (d) but not property (e), and τ_2 satisfies properties (d) and (e).

2. Consider the very simple rewrite theory in module `SIMPLE-SYS`. The reason for using such a simple example in an exercise is that you can *draw it* as an (unlabeled) graph, and you can also inspect the graph to see if some LTL properties hold when atomic propositions are added to it.

   Hint: You may find it quite helpful to draw such a graph.

   We can associate to this rewrite theory a Kripke structure on the set of atomic propositions $AP =$ `{a,b,c}` by equationally defining its labeling function in the module `SIMPLE-SYS-PREDS`. Then, by also importing the `MODEL-CHECKER` module you can verify some of its properties.

   In particular, write in LTL and verify by model checking that the following properties are true *from the initial state* `s0` (convince yourself that each of these properties is true by inspecting the graph, where next to each state you can write in another color the atomic propositions that are true in that state):

   (a) If after one step `b` holds, then eventually `b` or `c` always holds.

   (b) If after one step `b` holds, then after two steps from the initial state `a` holds until `b` or `c` always holds.

   (c) If after one step `c` holds, then after two steps from the initial state `a` always holds.

   (d) Eventually, either `a` always holds, or `b` or `c` always holds.

   (e) Eventually, either `a` always holds, or `b` leads to `c`.

   (f) Eventually, either `a` always holds, or `c` leads to `b`.

   (g) Eventually, either `a` leads to `a`, or `b` leads to `c`.

   (h) Eventually, either `a` leads to `a`, or `c` leads to `b`.

Hint: Since these formulas nest several temporal logic operators, *unless you enclose some subformulas in parentheses*, the Maude parser may interpret your formula in a different way than the one you mean. For example, `[] a U b` may mean one of two different formulas: either `([] a) U b`, or `[] (a U b)`. You should avoid any such ambiguities by adding enough parentheses.

```
mod SIMPLE-SYS is
  sort System .

  ops s0 s1 s2 s3 s4 s5 s6 : -> System [ctor] .

  rl s0 => s1 .
  rl s1 => s2 .
  rl s2 => s3 .
  rl s3 => s4 .
  rl s4 => s3 .
  rl s0 => s5 .
  rl s5 => s6 .
endm

load model-checker.maude

mod SIMPLE-SYS-PREDS is protecting SIMPLE-SYS .
  including SATISFACTION .
  subsort System < State .
  ops a b c  : -> Prop .

  eq s0 |=  a = false .  eq s0 |=  b = false .  eq s0 |=  c = false .
  eq s1 |=  a = false .  eq s1 |=  b = true .  eq s1 |=  c = false .
  eq s2 |=  a = true .  eq s2 |=  b = false .  eq s2 |=  c = false .
  eq s3 |=  a = false .  eq s3 |=  b = true .  eq s3 |=  c = false .
  eq s4 |=  a = false .  eq s4 |=  b = false .  eq s4 |=  c = true .
  eq s5 |=  a = false .  eq s5 |=  b = false .  eq s5 |=  c = true .
  eq s6 |=  a = true .  eq s6 |=  b = false .  eq s6 |=  c = false .
endm

mod SIMPLE-SYS-CHECK is
protecting SIMPLE-SYS-PREDS .
including MODEL-CHECKER .
endm

*** give here your eight model checking commands in the order listed above
```