CSC410

ASSIGNMENT 3

LTL Problems

Problem 1 – Warmup (12 points)

Alice and Bob work together in a workshop and share some tools. They also have different set of skills and may need each others' help to finish a project from time to time. Consider the following atomic propositions:

- at: Alice is using tool t.
- bt: Bob is using tool t.
- aw: Alice has requested Bob's help with a task and is *waiting* for help.
- bw: Bob has requested Alice's help with a task and is *waiting* for help.
- ah: Alice provides help to Bob.
- bh: Bob provides help to Alice.

Translate each of the following desirable properties of the workshop to an LTL formula:

(a) Whenever Alice is waiting for Bob's help, Alice is not using the tool t.

(b) If Bob requests help, then either Alice will eventually help or Bob will be waiting forever.

(c) If Alice requests Bob's help, then Bob cannot request Alice's help until he provides help to Alice.

(d) Neither Bob nor Alice can request the other person's help if their help has already been requested until they provide the requested help. To clarify: this is like adding a "vice versa" to (c) so that it states the same condition about Bob requesting help from Alice.

Problem 2 (20 points)

Below, use atomic propositions p_1, p_2, and p_3 to indicate that processes 1, 2, and 3 are respectively being executed by the scheduler. It is OK to assume that the scheduler can run a single process at any given time. Use req to indicate that a request is issued, and res for a response being issued.

(a) **Operating Systems**: processes 1, 2, and 3 are executed by a round-robin scheduler. That is, each process gets executed for one time step, and then we switch to the next.

(b) Modify the above to let the scheduler run each process for more than one step at a time, but only for finitely many steps before switching to the next process.

(c) **Client-Server Systems**: a response is only ever issued if there is a request pending. That is, no response is issued if there is no request or if the request has already received a response.

Below, assume that the light bulb can be of colours white (w), red (r), green (g), and blue (b) when it is on, or it can be off (o). The system changes state every second, and the status of the light accordingly changes (or remains the same). Note that it is assumed that the light is always exactly one colour.

Translate each of the following English specification for this simple system to an LTL formula.

(d) The light bulb turns red *at most* once. To clarify: turning red means the bulb should be a different colour in the previous step and change colour to red. It can then stay red for an arbitrary amount of time. But, if it changes colour to something else, then it cannot turn back to red again.

(e) The light bulb can only turn white if it has previously been at least once blue, once green, and once red (but not necessarily in that order).

Problem 3 (24 points)

Which one of the following equivalences hold? Give a formal proof for the correct ones and provide a counterexample for the incorrect ones. A counterexample is an infinite path that satisfies one side and not the other. You may not use any of the equivalences from the lecture/book as a boost. You are meant to prove these from scratch whenever they hold.

(a) $\varphi \,\mathtt{U}\, \neg\varphi \equiv \mathit{true}$

(b) $(\Diamond\, \Box\, \varphi_1) \wedge (\Diamond\, \Box\, \varphi_2) \equiv \Diamond(\Box\varphi_1 \wedge \Box\varphi_2)$

(c) $\Box\Diamond\varphi \implies \Box\Diamond\psi \equiv \Box(\varphi \implies \Diamond\psi)$

(d) $\varphi \,\mathtt{U}(\psi \vee \neg\varphi) \equiv \Box\varphi \implies \Diamond\psi$

Problem 4 (10 points)

Recall that satisfiability and validity of LTL formulas are defined in the same way as propositional logic formulas. An LTL formula φ is **satisfiable** if and only if there exists a path π that satisfies it ($\exists\pi : \pi \vDash \varphi$). An LTL formula φ is valid if and only if all paths π satisfy it ($\forall\pi : \pi \vDash \varphi$). Note that validity of φ can also be reformulated as the equality $\varphi \equiv \mathit{true}$.

For the formulas below, determine if the formula is satisfiable, unsatisfiable, or valid. Formally justify your answer.

(a) $\Diamond b \implies (a \,\mathtt{U}\, b)$.

(b) $\bigcirc(a \vee \Diamond a) \implies \Diamond a$

CTL Problems

Problem 5 (16 points)

Recall the setup of Problems 1 and 2. We will reuse them for this problem to write a few more properties in CTL.

(a) Bob cannot ask for Alice's help unless he has already helped Alice at least once. Note that Bob does not have to ask for Alice's help at all; but, if he does, it should be after having helped Alice before.

(b) If Alice asks for Bob's help, then it is a future possibility (but not necessity) that the tool remains available from this moment (that the help was requested) until the help is delivered.

(c) The light bulb has a possible future in which it is never indefinitely stuck on any one colour.

(d) If the light bulb has ever switched from white to blue in the past, then it cannot switch from blue to white in the future.

Problem 6 (16 points)

Let TS be a finite transition system (over AP) without terminal states (i.e. every state has an outgoing transition), and Φ and Ψ be CTL state formulae (over AP). Prove or disprove: $TS \vDash \exists(\Phi \,\mathtt{U}\, \Psi)$ if and only if $TS' \vDash \exists\Diamond\Psi$ where TS' is obtained from TS by eliminating all outgoing transitions from states s such that $s \vDash \Psi \vee \neg\Phi$.

Problem 7 (20 points)

Which one of the following equivalences hold? Give a formal proof for the correct ones and provide a counterexample for the incorrect ones. Proofs should be from scratch, and not by referencing other equalities.

(a) $\forall \Box \exists \Diamond (\varphi_1 \wedge \varphi_2) \equiv \forall \Box \exists \Diamond \varphi_1 \wedge \forall \Box \exists \Diamond \varphi_2$

(b) $\exists \Box \forall \Diamond (\varphi_1 \wedge \varphi_2) \equiv \exists \Box \forall \Diamond \varphi_1 \wedge \exists \Box \forall \Diamond \varphi_2$

(c) $\exists \Box \varphi \equiv \varphi \wedge \exists \bigcirc \exists \Box \varphi$

Model Checking

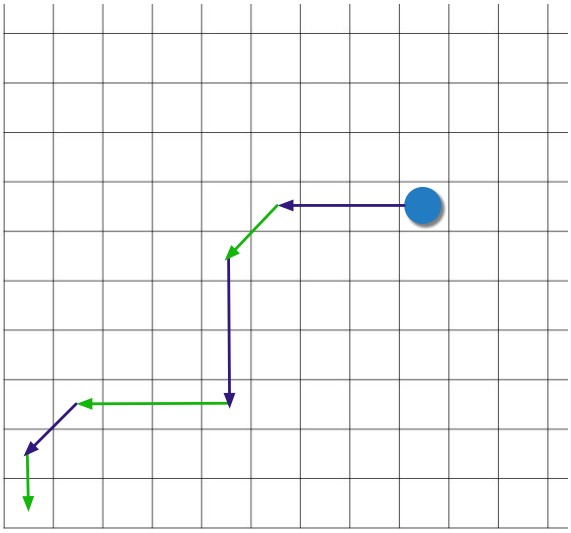
One more problem left on model checking, which will be released in synch with class on November 23rd. The grad problem will also be released by that date.

Problem 8 (30 points)

This problem is for undergraduates only, and its goals is to ensure the understanding of the ideas behind the CTL model checking algorithm, specifically the way universal and existential *until* is computed through fixpoints.

There is a game played on a grid of squares with one piece which is initially located at a position (m, n) of a grid (with $m, n \in \mathbb{N}$). The grid's origin $(0, 0)$ is at the bottom left corner and it is arbitrarily large including all squares with pairs of natural number coordinates.

The game is played between two players, who take alternate turns to move this game piece towards the origin. The valid moves for the piece are like a chess queen, as long as the direction of the move is towards the origin, i.e. left, down or diagonally towards left-down. Like a chess queen, the piece is allowed to travel as far as the player chooses in a valid direction during the one move. The player that moves the piece to the origin wins the game. Below is an example play of the game:



played from the initial location $(8, 6)$ where the first player loses the game.

We say a player has a *winning strategy* for a game iff there is play for this player to win this game independent of the choices that the opponent makes. For example, the first player always has a winning strategy from any location $(n, 0)$, $(0, n)$, or (n, n) because the player can move the piece in one move to the origin and win.

A two-player game is called *determined* if from any given position, exactly one of the players has a winning strategy. The above game is determined. Given two excellent players and any location (m, n), either player one always wins the game from (m, n) or he always loses.

The goal of this exercise is to implement a *decision procedure*. The input will be the pair of numbers (m, n). The output is "1" if the first player has a winning strategy from this location, and "2" if the second player has a winning strategy from this location.
Note: this problem is not a random implementation problem. To come up with a solution that scales up, you are encouraged to think carefully about how checking for the existence of a winning strategy relates to the concepts of existential and universal path properties. You are also encouraged to think about the algorithm we discussed for *until* and how the idea behind that algorithm can hint at a nice solution for this problem.
Forbidden Implementation Tricks: In order to let your solution *win* on merit, rather than hacks, we explicitly forbid any sort of optimization trick. For example:

- You cannot boost your solution by giving it a lookup table for smaller values. For example, one can manually computer all solutions for a 10×10 grid, enter those values as a constant for the algorithm, and let further points to get to one of these points. **This will be considered cheating.**
- You cannot use an oracle like a SAT/SMT solver under the hood.

The list above is naturally not comprehensive, because it is impossible to a priori guess any trick you may have up your sleeves as an intelligent bunch. But, it should give you the idea that we are not looking for shortcuts, but elegant algorithmic solutions.