Assignment 5

Due on Friday November 22nd, 2019 at 11:59pm

Assignment Format and Guidelines on Submission

This assignment is worth 15% of the total course mark. Submit a properly typed PDF on Markus. No handwritten assignment will be accepted. Unless otherwise specified, no English words will be marked.

This assignment will be released gradually in parts. Some of the material will be released in the week after the break when we finish the model checking topic. You are strongly advised to finish the parts that are released early, so that you have time for the remaining parts when they are released. If you do not pace yourself, this will become a big hassle for you near the deadline.

This assignment may look long (at the end), but keep in mind that it is designed for a group of 3 or 4 students and you have collectively over 20 days to get it done. The volume is adjusted accordingly.

List of files to submit:

• a5.pdf which will include the cleanly typed solutions to problems 1-6.

• source.zip which will include the solution to problem 7 as described in the text of the problem.

This assignment is now complete!

Problem 1: Symbolic Testing

(20 points) Consider the following function foo:

```c
void foo(int x, int y){
    if (x + y > 10){
        x = x - 5;
        y = y + 5;
    } else {
        z = x;
        x = y;
        y = z;
    }
    if (x + y < 0){
        x = -x;
        y = -y;
    } else {
        x = -x;
        y = -y;
    }
    assert(x + y > 0)
}
```

(a) Enumerate all the paths in foo where each path is given as a sequence of line numbers.

(b) Are all the paths listed above feasible? Prove the paths are not feasible as such by filling instances of a table in the below format. At the end of each, write a short English description to sum up the result of the table.

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Assignment</th>
<th>Path Condition</th>
</tr>
</thead>
</table>

1
(c) Of all the feasible paths, would any result in an assertion violation? Prove your claim using symbolic execution. Fill in a table in the above format again and conclude your argument using a short English description.

LTL Problems

Problem 2

(15 points) Let us assume we have a system with only one observable component: a colour LED light bulb. This 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 on 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.

(a) When the light bulb turns blue, it means the system is in low battery mode and cannot turn the bulb into any other colour. It will remain blue for while until the remainder of the battery runs out and the light turns off forever.

(b) The light bulb stays on forever, changes colour every second, and alternates between red and white.

(c) The light bulb stays on forever, alternates between colours red, blue, and white (in that order), staying at each colour for an arbitrarily long (but finite length) amount of time.

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

(e) The light bulb turns red exactly once (but it can remain red for any duration).

Problem 3

(30 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.

(a) $\varphi \mathcal{U} \neg \varphi \equiv true$

(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) $\varphi \mathcal{U} (\psi \vee \neg \varphi) \equiv \Box \varphi \implies \Diamond \psi$

(e) $\Diamond \Diamond \varphi \equiv \Diamond \varphi$

Problem 4

(10 points) We can define satisfiability and validity of LTL formulas the same way that we do for propositional logic formulas. An LTL formula $\varphi$ is satisfiable if and only if there exists a path $\pi$ that satisfies it ($\exists \pi : \pi \models \varphi$). An LTL formula $\varphi$ is valid if and only if all paths $\pi$ satisfy it ($\forall \pi : \pi \models \varphi$). Note that validity of $\varphi$ can also be reformulated as the equality $\varphi \equiv true$.

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

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

(b) $\Diamond (a \vee \Diamond a) \implies \Diamond a$
CTL Problems

Problem 5

(15 points) Recall the setup of Problem 2 with the LED light bulbs. We will reuse it for this problem to write a few more properties in CTL.

(a) If the light bulb is white, then it is possible for it to turn blue in the future, but if it is green then it will never turn blue in the future.

(b) Once the light bulb is red, then it has to remain red until it is turned off, unless it is turns green (from red) which cancels the red out.

(c) The light bulb 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.

(e) The light bulb turns green exactly twice (but can remain green for any duration each time).

Problem 6

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

Model Checking Problems

Problem 7

(45 points) The goal of this problem is to ensure that you understand the ideas behind the CTL model checking algorithm, and 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.

Format

You are free to implement this in the programming language of your choice. Submit your source files as one zipped directory source.zip. This directory should include an executable called game that runs on the CDF machines. The input is passed to your executable as a command line parameter, that is:

./game 2 1

should execute on a CDF machine and return “2”, since the second player has a winning strategy from the location (2, 1).

Grading

There is a naive algorithm to solve this which will scale very poorly. What does poorly mean? It means that the algorithm has exponential complexity and will likely take over a minute to process a location as small as (12, 10). You may assume that this naive algorithm will get no marks. The reason is that this default naive solution is something anyone who know programming can implement and has nothing to do with the material taught in this class.

A reasonable algorithm should handle the same location (12, 10) in a small fraction of a second. It would be imprecise to put an exact number on this (since it will be hardware dependent), but think of it as around 0.01s. But, more importantly, you should not see a substantial jumps for small coordinate changes at these values, for example, between the times for (12, 10) and (13, 12). As an another example, think about your algorithm scaling up to around coordinates (60, 60) with the execution time remaining under one minute. A solution like this will take the full mark.

Finally, you can be extra clever about your implementation and beat the above time by an order of magnitude or two. For example the time for any point in the 60 × 60 grid will be under one second. We will give a special bonus to these solutions. Consider this as a way to compensate for a lower mark on Exam 2. We will give any group with a solution like this an extra 15 points on this assignment. This can translate to an increase of 1.5 points in your total course mark. Of all these groups, the group that beats everyone else by a clear margin will be granted a bonus of 20 points. The collection of the bonus points will be conditional on a representative of the group presenting their solution to the class. Depending on the timing, this will either take place during our last class or on Piazza.

Note that we will not grade your source code. We ask you to submit it for insurance, that is, in case something goes wrong with the executables and you would like to reclaim your mark through the original material submitted.