CS 6110, Spring 2022, Assignment 5

Given 2/22/22 – Due 3/3/22 by 11:59 pm via your Github

NAME: **UNID:**

CHANGES: Please look for lines beginning with underlined words when they are made. none yet.

Answering, Submission: Have these on your private Github: a folder Asg4/ containing your submission, which in detail comprises:

- A clear README.md describing your files.
- Files that you ran + documentation (can be integrated in one place).
- A high level summary of your cool findings + insights + learning – briefly reported in a nicely bulletted fashion in your PDF submission.

Start Early, Ask Often! Orientation videos and further help will be available (drop a note anytime on Piazza for help).

I encourage students constructing answers jointly! *But that does not mean copy solutions, but discuss the question plus surrounding issues.*

1. (10 points) Read and summarize Jackson's article on Alloy from `https://cacm.acm.org/magazines/2019/9/238969-alloy/fulltext#R1` in exactly two pages. Please capture all the details highlighted there. Try to write a good summary for your own understanding! You may include a few diagrams in your 2-pager (hence, revised this question to "2 pages"). Note: here is an embedded video on this page that is very helpful!

Your Answer Here

2. (25 points) Take the tutorial on `https://www.doc.ic.ac.uk/project/examples/2007/271j/suprema_on_alloy/Web/index.php` and take the assessment questionnaire at the end! Try to score a perfect score (it gives you hints and retry opportunities). *Record that you did the above in about two pages – capture screenshots, sessions, etc, convincing me that you took the assessment.* Report your assessment score as a screenshot! (Get a perfect 10/10.)

3. (40 points) First, deep-read the Alloy "Red book" linked on the class website from the beginning till Page 80. Summarize the concepts you find on these pages in four pages. *I want to see a good summary.*

   Then do the work below. Three models for trees are below (these are from `https://stackoverflow.com/questions/41707898/is-there-a-better-alloy-model-of-a-tree`). I saw the beginnings of Costello's tree solution (which Daniel Jackson, author of Alloy, improves upon) and wrote my own solution. Thus, there are three tree models below. Study them and do the problems stated below them:

```
/*-- BEGIN: common to all three tree definitions --*/

sig Node {
  tree: set Node
}

one sig root extends Node {} // root is a subset of Node
```

```
/*-- END : common to all three tree definitions --*/

pred GGTree{
 no n : Node | root in n.^tree // Q-1
 --
 no n : Node | n in n.^tree    // Q-2
 --
 all n : Node - root | n in root.^tree // Q-3
 --
 all n : Node |
 all disj n1, n2 : n.tree |    // Q-4
   no (n1.*tree & n2.*tree)
}
```

Q-1,2: Express what these definitions are saying, in English. Why can't these definitions use $*$? What happens if you do that? Generate a few tree instances (models) and show what happens if you use $*$ for Q-1 and Q-2?. *To generate instances,* you must remove the "pred" temporarily, and express its contents as signature facts. This is what you would be doing initially in your work on creating a model, anyhow. Then follow such a modified definition with a `run {} for 3`, or `run {} for exactly 10 node` or similar notations (consult the "Red" Alloy book mentioned on the class website for details). Then execute, and the "run" gives you the instances I'm looking for.

Q-3: Express what this definition is saying, in English. Why do we need `Node - root`? Why can't you use `Node` here? Again show by generating instances saying what happens if you do that.

Q-4: What is this saying in English? What happens if you leave out `disj`? Show by generating some instances.

- Now put the `pred` definitions back. So you have three "pred" definitions as presented below.

```
pred GGTree{
 no n : Node | root in n.^tree
 --
 no n : Node | n in n.^tree
 --
 all n : Node - root | n in root.^tree
 --
 all n : Node |
 all disj n1, n2 : n.tree |
   no (n1.*tree & n2.*tree)
}


pred DJTree {
    Node in root.*tree // all reachable
    no iden & ^tree // no cycles
    tree in  Node lone -> Node // Q-5
    }

pred CostelloTree {
    // No node above root (no node maps to root)
    no tree.root
    // Can reach all nodes from root
```

```
    all n: Node - root | n in root.^tree
    // No node maps to itself (irreflexive)
    no iden & tree
    // No cycles
    no n: Node | Node in n.^tree
    // All nodes are distinct (injective)
    tree.~tree in iden -- need this
}
```

Explain the line tagged Q-5, above. You will find the constraints of signatures explained in the "Red" book. Look around Page 78, Section 3.6.3.

The goal is to show the equivalences below:

(a) Issue `check {GGTree iff DJTree} for 5` and see if the definitions are equivalent.

(b) Just break the Q-1 to have $*$ and not the correct `^`. Do the "`iff`" check above. Do you get an understandable counterexample? Describe it.

(c) Issue `check {CostelloTree iff DJTree} for 5` and see if the definitions are equivalent.

Your Answer Here

4. (25 points) Write your own quicksort in C for a character array of 5 (five) locations, and subject it to KLEE tests, as illustrated in Lecture 13. Write a bug-free version. Then put one bug that does not sort correctly. Put an assertion at the end for a mis-sorted outcome. Hit that assertion using a KLEE-test.

Your Answer Here

Piazza posting:

```
PIAZZA posting:

I changed 8 to 5 in the HW specification (with 5 itself, it generates 120 tests).
With 8, it takes forever. Why torture you?

Also try and remove printf etc before generating tests.

One can write a script to automatically run for all the synthesized tests.

For "breaking" qsort, you can do it in a certain way (to make our life easy,
I'll suggest one):

1) make the pivot = the first element of the incoming array

2) Write a broken qsort by checking the pivot first. If the pivot is < 5 (say),
break the qsort (let it return w/o sorting). For >= 5, make the qsort sort the array

3) Try it and keep telling me what you find out.

This shows that symbolic execution of C is not "that free" - yet highly automated.
Also I'll read-up on how to declare larger arrays and maybe make the elements
selectively symbolic (only some symbolic).
```

Ganesh