Dear students,

Below are the most frequently asked questions of this week's lab (week 4):

Question 1: The solver is returning a lot of empty strings as a trace, what am I doing wrong?

Answer:

It could be that you did not add the constraint that the input value returned by the solver must be a character from the input symbols of the given problem. The code for this constraint is listed in last year's FAQ but to make it easier for you, we will also list it here:

```
BoolExpr constraint = c.mkFalse();
for (String input: PathTracker.inputSymbols) {
    constraint = c.mkOr(c.mkEq(z3var, c.mkString(input)), constraint);
}

PathTracker.addToModel(constraint);
```

Add this code right after the line where you have initialized "`z3var`".

Furthermore, remember to add the newly created MyVar for the input to `PathTracker.inputs`! Forgetting to add it to "`PathTracker.inputs`" would mean it is not taken into account when the solver is trying to solve the path constraint and you will get empty strings back from the solver.

Question 2: To create the expression to the opposite branch, we used "`mkNot(condition)`". Is this correct?

Answer:

Do not directly use "`mkNot`" on the condition as this would actually change the expression of the condition instead of what the condition evaluates to; you are technically solving for a branch that does not exist on the same line. We understand that the return value of this method is a BoolExpr, but the effect is different e.g. say we have "a==b" as the condition of the if-statement that we just triggered and we are in the false-branch. Using the "mkNot" method would turn it into "a != b". But this is a completely different expression than the actual condition of the if-statement! You want to ask the solver to find inputs that could lead you to the true-branch (so where "a==b" is true and not that "a != b").

Question 3: How do we know whether a path constraint is not satisfiable?

Answer:

You can make a call to "`PathTracker.solve()`" to the opposite branch. If this is unsatisfiable, the newSatisfiableInput will not be called. You can use a boolean flag to get the result back in the encounteredNewBranch like this:

```
private static boolean satisfiable = false;

    static void encounteredNewBranch(MyVar condition, boolean value, int
line_nr){
        satisfiable = false;
        // Call the solver
        PathTracker.solve(...);

        if(satisfiable) {
            // SATISFIABLE
        } else{
            // UNSATISFIABLE
        }
    }

    static void newSatisfiableInput(LinkedList<String> new_inputs) {
        satisfiable = true;
        // Hurray! found a new branch using these new inputs!
    }
```

With this solution, we assume that if we don't hear from the solver, we take the branch to be unsatisfiable.

Another solution is to update the PathTracker.solve() method to return a boolean that indicates satisfiability.

Question 4: If we use "`PathTracker.ctx.mkEq(condition.z3var, mkBool(!value))`", we only get unsatisfiable as output. What are we doing wrong?

Answer:

We suspect that if you use "`mkBool(!value)`", it will return a more generic type of a `BoolExpr` which is hard for the solver to know whether it should be true or false. We suggest using "`mkTrue()`" and "mkFalse()" instead.