PROGRAM ANALYSIS, VERIFICATION AND TESTING

(CS639)

ASSIGNMENT - 2

Submitted by:

PRATIBHA GUPTA
231110038

In the given task we are supposed to implement Symbolic Execution. For which we need to complete the function in the ‘symbSubmission.py’ file which is at the location ‘Chiron-Framework\Submission’. The details of implementation of the functions are given as:

IMPLEMENTATION -

We are given two Kachua programs, in which one program has holes in the form of constant parameters along with input values and in other programs there are no constant parameters. We are supposed to find the values for constant parameters in the program with the holes so that this program becomes semantically equivalent to the program with no constant parameters.

So first we generate JSON files for both the programs, this file contains the value of input parameters, constant parameters, path constraints for the path taken and information about the programs. Here, “testData1.json" is the file for the program with constant parameters and “testData2.json” is the file for the program without constant parameters.

I have created four z3 solvers:

Solver s - is used to solve the constraints from testData2 with the input values and gather the conditions for each path constraint.

Solver s2 - is used to solve the symbolic encodings between program 1 and program 2 when there are no constant parameters in the path constraint.

Solver temp - is used to compute the values of constraints using the constant parameters.

Solver s3 - is used to solve the symbolic encodings between program 1 and program 2 when there are constant parameters in the path constraint.

1- Using the testData, using “params” and “constparams” values I have added the required symbols in the solver.
2- Now for each input I have checked which path constraints from testData1 are satisfied. For that constraint I have mapped symbolic encodings, there are no constant parameters
3 - When there are constant parameters in the constraint, I have created a temp solver to solve the constraints with the constant parameters from testData1 and then for these constraints I have mapped the symbolic encoding from both json files.
4 - After creating the solver s2 or s3 I gave it the z3 solver which returns the value of constant parameters such that both programs are semantically equal.

ASSUMPTION -

1- We need to name the json file for a program with constant parameters as “testData1.json” and another one as “testData.json”.

2- We are required to give the same inputs to both the programs while creating the json files.

LIMITATIONS -

1- For a large program it is possible that the time allotted is not sufficient to explore all the paths. In this case the semantic equivalence can’t be established.