Program Analysis Verification and Testing Assignment-2

Name:- **Govind Sharma**

Roll-No:- **231110015**

Department:- **MTech-CSE**

Introduction:

Project Purpose and Objectives:

In this assignment we are comparing the behaviour of two programs after analysing it, to determine if they are semantically equivalent for a given test cases. The project involves the following key objectives:

Symbolic Execution: Using symbolic execution techniques to generate test data and execution traces for program and producing a "testData.json" file.

Check Equivalence: Implementing the "checkEq()" function in the "symbSubmission.py" file which is the main portion of the whole assignment which is finding the assignments to constant parameters stored in "testData.json" that will make given programs equivalent for specified test cases.

Generating .kw Files: Generate ".kw" files from ".tl" files for program analysis using the "chiron.py" script.

Program Equivalence: Develop constraints and logic to find values for constraints such that, for all test cases, program will produce the same output values for specified variables (e.g., "x" and "y").

Implementation:

Imports: The code is importing necessary Python modules and external libraries. These include z3, argparse, json, sys, sExecutionInterface, z3solver, irgen, interpreter, ast.

example Function: This function takes three arguments: s (an instance of z3Solver), constrinats_dict (a dictionary of constraints), and variables_set (a set of symbolic variables). It appears to set up and solve symbolic constraints using the Z3 solver.

Inside this function, symbolic variables are added to the solver using the addSymbVar method for each variable in variables_set.

Then, constraints are added to the solver by iterating over constrinats_dict. The constraints are built as strings and added using the addConstraint method.

Then Z3 solver is used to check the satisfiability of the constraints using s.s.check(). If the result is "sat" (satisfiable), the constraints are printed.

The checkEq function compares two JSON files containing program test data. It uses the Z3 solver for symbolic execution and constraint solving. The data is processed via nested loops to extract symbolic variables and constraints. It specifically stores constraints related to the variable 'y' in constraints_dict. After processing, the code prints the collected constraints and variables. Command-line arguments are parsed using argparse, with the program file as a required input and optional flags like -b (binary IR loading) and -e (variable specification).

Main Execution: The if __name__ == '__main__': block is the entry point for the script. It parses the command-line arguments, loads the Intermediate Representation (IR) from the specified program file, and then calls the checkEq function to perform the equivalence checking.

Limitations: This python program is only able to run on limited number of variables and Dependent on external modules or custom functions that are not provided.