Implementation Report: Program Synthesis using Symbolic Execution

Neelu Lalchandani
231110031

October 5, 2023

1 Introduction

Symbolic Execution tool is designed to find constant assignments to variables in program P_1 such that it becomes semantically equivalent to program P_2. This report outlines the implementation, assumptions, and limitations of the tool.

2 Implementation Details

The tool is implemented in Python and relies on the Z3 solver for symbolic execution and constraint solving. It comprises several modules, including sExecutionInterface.py, z3solver.py, irgen.py, interpreter.py, and ast.py, each responsible for specific functionalities like symbolic variable addition, constraint handling, and symbolic encoding.

P1 program has holes/constant parameters and P2 program is a complete turtle program. We get input variables and output equations from P2, then we get constraints from P1 and output equations too.

2.1 Program Description

- Program P1 has holes/constant parameters that need to be filled in.
- Program P2 is a complete turtle program that we aim to make P1 equivalent to.

2.2 Tool Workflow

The tool's workflow involves:

- Extracted input variables and output equations from P2.
- Collected constraints from P1.
- Gathered output equations from P1.

- Created two solvers
- One for checking if the variables satisfy the constraints
- If yes, then we add the symbolic encoding of equal equations from P1 and P2 and add them to second solver
- The second solver finds the values of constants for which programs are equal

3 Assumptions

The tool operates under the following assumptions:

- The tool assumes that programs P1 and P2 are provided in a specific format compatible with the tool's interface, and the functions example(s) and checkEq(args, ir) are appropriately defined.
- It assumes that the symbolic variables, constraints, and assignments are specified using the correct syntax as demonstrated in the provided code snippet.
- The tool assumes that the input programs are free from syntax errors and conform to the expected format.

4 Limitations

The tool has several limitations:

- This implementation works for only a few variables (maximum 12) and a limited number of holes (maximum 7). This tool can also be modified further to work with many variables.
- The tool may not handle certain complex program structures or specific types of constraints effectively. For example, intricate loop structures may lead to incomplete or incorrect results.
- It may struggle with large-scale programs or those with a high number of variables, as the symbolic execution process can be computationally expensive.
- The tool may not provide meaningful results for programs with nonlinear or highly complex arithmetic operations.
- Additionally, if the input programs contain unsupported language features or constructs, the tool may fail to produce accurate results.
- In some cases, the tool may not provide a solution even if one exists due to limitations in the symbolic execution process.