Student Name: Nitish Kumar
Roll Number: 231110033
Date: October 5, 2023

INTRODUCTION

The report is for the "Program Synthesis using Symbolic Execution". In this assignment, the primary objective is to find constant assignments to variables in a given program **P1**, such that **P1** becomes semantically equivalent to **P2** where **P1** contains some hole(constraints) and **P2** have the exact values available. This task involves utilizing symbolic execution and the Z3 theorem prover to identify the necessary constant assignments that establish program equivalence.

Workflow of the codes submitted

1. **Import Dependencies:**
   - The code starts by importing the necessary libraries and modules, including Z3 for symbolic execution, argparse for command-line argument parsing, json for JSON file handling, and custom modules related to symbolic execution and program interpretation.

2. **Define example() Function:**
   - A sample function **example()** is defined to demonstrate how to use the Z3 solver. This function adds symbolic variables, constraints, and assignments to the solver and provides examples of interacting with the solver.

3. Workflow of the checkEq() Function:

i. **Opening JSON Files:**
   - The function starts by opening two JSON files, **testData2.json** and **testData1.json**, for reading and parsing. These JSON files presumably contain data generated during symbolic execution of the two programs P1 and P2.

ii. **Creating a Z3 Solver Instance:**
   - Next, an instance of a Z3 solver is created using the custom module **zs.z3Solver()**. This solver will be used to handle symbolic variables and constraints.

iii. **Iterating Through Data Pairs:**
   - The function enters a loop that iterates through pairs of keys, **key1** and **key2**, where **key1** comes from the first JSON file (**testData10.json**) and **key2** comes from the second JSON file (**testData9.json**).
   - The pairs of keys are generated based on the condition that the parameter values (**'params'** field) in the two JSON files match. This condition ensures that the code compares data entries with the same parameter values.

iv. **Printing Data:**
   - Inside the loop, the code prints various pieces of data from both JSON files. These include parameter values and symbolic encodings.
   - The purpose of these print statements is likely for debugging and verification of the data being processed.

v. **Extracting Data from JSON:**
   - The function extracts the following data for each pair of keys:
     - **j_params_s1** and **j_params_s2**: The parameter values as JSON strings from the two JSON files.
     - **j_s1** and **j_s2**: The symbolic encodings as JSON strings from the two JSON files.

vi. **JSON Data Parsing:**
- The code performs JSON data parsing by replacing single quotes (**'**) with double quotes (**"**) in the JSON strings obtained from the JSON files.
- It loads these modified JSON strings into dictionaries named **dict1**, **dict2**, **dict3**, and **dict4**.

vii. **Adding Symbolic Variables:**
- The code iterates over the keys in **dict3**, which likely represent symbolic variables.
- For each symbolic variable, it adds the variable to the Z3 solver using the **s.addSymbVar(t)** function. This step ensures that the solver is aware of the symbolic variables.

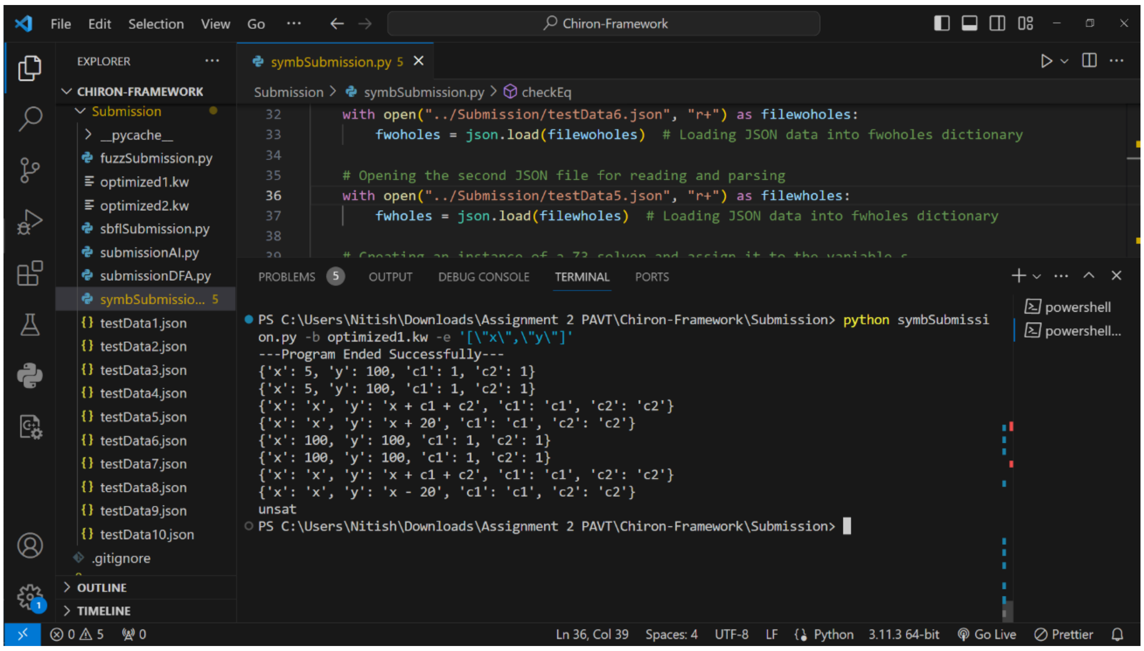
viii. **Adding Constraints:**
- For each symbolic variable, the code adds a constraint to the Z3 solver using the **s.addConstraint(f"{dict1[t]}=={dict2[t]}")** function. These constraints express the requirement that the corresponding symbolic variables in P1 and P2 should be equal.

ix. **Solving Constraints:**
- After adding all the constraints, the code checks if the constraints are satisfiable by calling **s.s.check()**. The result of the solver's check is stored in the **res** variable.

x. **Printing Result:**
- The code prints the result of the solver, which can be either "sat" (satisfiable) or "unsat" (unsatisfiable).
- If the constraints are satisfiable (**"sat"**), it means that constant assignments exist to make P1 semantically equivalent to P2. In this case, the code prints the model obtained from the solver, which likely contains the constant assignments.

xi. **End of the Function:**
- Finally, the code prints a message indicating the successful end of the function, which serves as a confirmation that the function has completed its task.

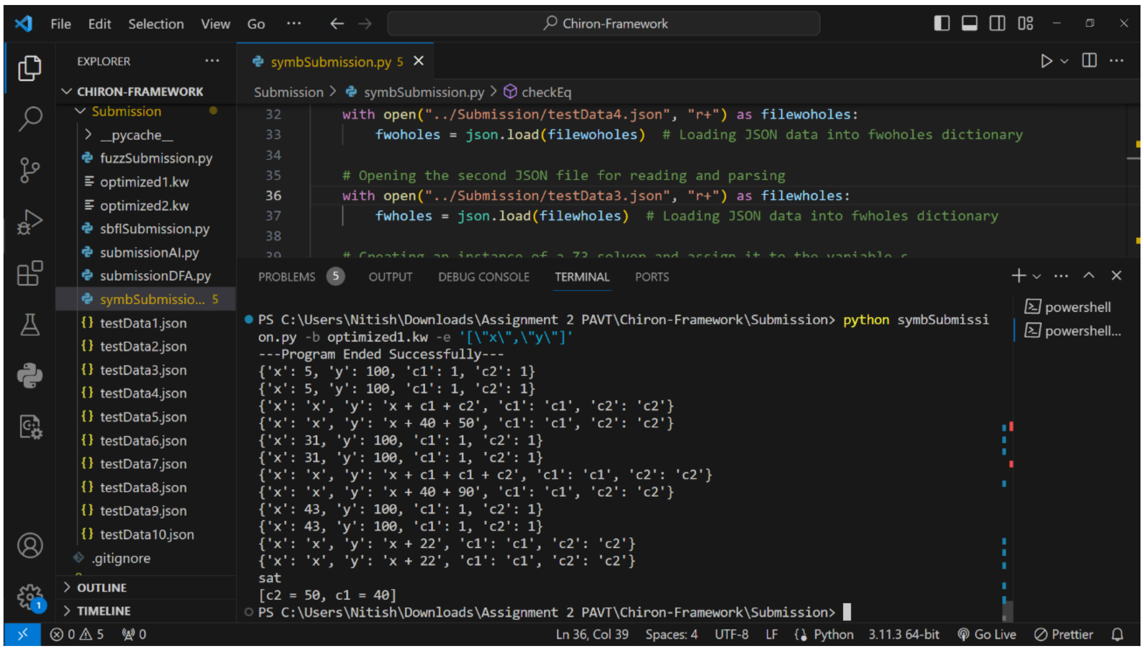
Images of some code snippets and outputs



```python
with open("../Submission/testData6.json", "r+") as filewoholes:
    fwoholes = json.load(filewoholes)  # Loading JSON data into fwoholes dictionary

# Opening the second JSON file for reading and parsing
with open("../Submission/testData5.json", "r+") as filewholes:
    fwholes = json.load(filewholes)  # Loading JSON data into fwholes dictionary
```

```
PS C:\Users\Nitish\Downloads\Assignment 2 PAVT\Chiron-Framework\Submission> python symbSubmissi
on.py -b optimized1.kw -e '[\"x\",\"y\"]'
---Program Ended Successfully---
{'x': 5, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 5, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 'x', 'y': 'x + c1 + c2', 'c1': 'c1', 'c2': 'c2'}
{'x': 'x', 'y': 'x + 20', 'c1': 'c1', 'c2': 'c2'}
{'x': 100, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 100, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 'x', 'y': 'x + c1 + c2', 'c1': 'c1', 'c2': 'c2'}
{'x': 'x', 'y': 'x - 20', 'c1': 'c1', 'c2': 'c2'}
unsat
PS C:\Users\Nitish\Downloads\Assignment 2 PAVT\Chiron-Framework\Submission>
```



```python
with open("../Submission/testData4.json", "r+") as filewoholes:
    fwoholes = json.load(filewoholes)  # Loading JSON data into fwoholes dictionary

# Opening the second JSON file for reading and parsing
with open("../Submission/testData3.json", "r+") as filewholes:
    fwholes = json.load(filewholes)  # Loading JSON data into fwholes dictionary
```

```
PS C:\Users\Nitish\Downloads\Assignment 2 PAVT\Chiron-Framework\Submission> python symbSubmissi
on.py -b optimized1.kw -e '[\"x\",\"y\"]'
---Program Ended Successfully---
{'x': 5, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 5, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 'x', 'y': 'x + c1 + c2', 'c1': 'c1', 'c2': 'c2'}
{'x': 'x', 'y': 'x + 40 + 50', 'c1': 'c1', 'c2': 'c2'}
{'x': 31, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 31, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 'x', 'y': 'x + c1 + c1 + c2', 'c1': 'c1', 'c2': 'c2'}
{'x': 'x', 'y': 'x + 40 + 90', 'c1': 'c1', 'c2': 'c2'}
{'x': 43, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 43, 'y': 100, 'c1': 1, 'c2': 1}
{'x': 'x', 'y': 'x + 22', 'c1': 'c1', 'c2': 'c2'}
{'x': 'x', 'y': 'x + 22', 'c1': 'c1', 'c2': 'c2'}
sat
[c2 = 50, c1 = 40]
PS C:\Users\Nitish\Downloads\Assignment 2 PAVT\Chiron-Framework\Submission>
```