Assignment 1 Report

CS 639: Program Analysis Verification And Testing

Complete a fuzzing loop

September 16, 2023

Nitish Kumar
231110033
M.Tech(CSE)

INTRODUCTION

In Assignment 1(Complete a fuzzing loop), We had to Implement the marked functions and class interfaces in Submission/fuzzSubmission.py file as follows:

- def compareCoverage(curr_metric, total_metric)

- def mutate(input_data) # input_data type is InputObject(...)

- def updateTotalCoverage(curr_metric, total_metric)

Workflow of the codes submitted

My first function which was compareCoverage(curr_metric, total_metric)

The **compareCoverage** function takes in two lists, **curr_metric** and **total_metric**, and it's designed to check whether **curr_metric** is different from any of the elements in **total_metric**.

This is how it works:

1. It uses a nested comprehension to iterate over both **curr_metric** and **total_metric** elements simultaneously using **zip(curr_metric, total_metric)**.
2. For each pair of elements **(curr, total)**, it checks if all corresponding elements are equal. It does this by using **all(curr == total[i] for curr, total in zip(curr_metric, total_metric))**, where **total[i]** represents the elements in one of the lists in **total_metric**.
3. If **all** pairs of elements are equal (i.e., if the **curr_metric** is equal to one of the lists in **total_metric**), it returns **False**, indicating that coverage has not improved.
4. If it doesn't find any matching lists in **total_metric**, it returns **True**, indicating that coverage has improved.

My second function was mutate(self, input_data, coverageInfo, irList)

The **mutate** function takes three arguments: **input_data**, **coverageInfo**, and **irList**. Its purpose is to mutate the **input_data** and return a mutated version of it. This is how it works:

1. **Copying Input Data**: The function begins by making a copy of the **data** attribute from the **input_data** object and assigns it to the variable **m_data**. This step is essential to avoid modifying the original **input_data** object.
2. **Checking for Non-Empty Data**: It checks if **m_data** is not empty using the **if m_data:** condition. If **m_data** contains one or more key-value pairs, it proceeds with the mutation process; otherwise, it returns the original **input_data** unchanged.
3. **Selecting a Variable to Mutate**: If there are key-value pairs in **m_data**, the function selects a random variable to mutate from the keys of **m_data** using the **random.choice(list(m_data.keys()))** method.
4. **Generating a New Value**: Next, it generates a new random value within the range of -20 to 20 (inclusive) using **random.randint(-20, 20)**.
5. **Updating the Variable**: It updates the selected variable in **m_data** with the new random value.
6. **Creating a Mutated Input**: Finally, it creates a new **InputObject** with the mutated **m_data** and returns it as the result of the **mutate** function.

My third function was updateTotalCoverage(curr_metric, total_metric)

he **updateTotalCoverage** function is a Python function that takes two arguments: **curr_metric** and **total_metric**. Its purpose is to update the **total_metric** list if **curr_metric** is not already in it and if the coverage, as determined by the **self.compareCoverage** method, has improved.

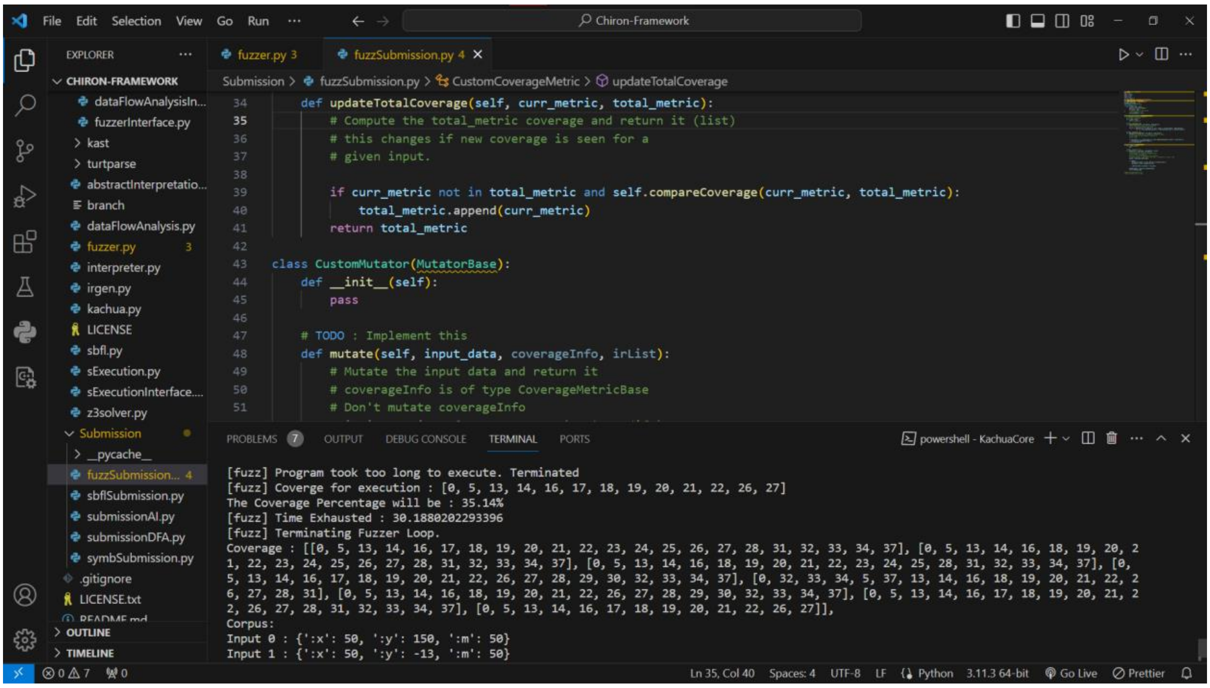
Here's a step-by-step explanation of how it works:

1. It checks if **curr_metric** is not already present in the **total_metric** list using the condition **if curr_metric not in total_metric**.
2. It also checks if the coverage has improved by calling the **self.compareCoverage** method with **curr_metric** and **total_metric** as arguments. The **self.compareCoverage** method likely determines whether **curr_metric** is different from all elements in **total_metric**.
3. If both conditions are met (i.e., **curr_metric** is not in **total_metric**, and the coverage has improved), it appends **curr_metric** to the **total_metric** list using **total_metric.append(curr_metric)**.
4. Finally, it returns the updated **total_metric** list.

I also added few lines to the fuzz.py file which is located in the KachuaCore folder

at line no. 126. That line is creating a string that combines a fixed text message with a dynamically calculated coverage percentage, rounded to two decimal places, and displays it in the form of a percentage. For example, if the coverage percentage is 75.25%, the line will print: "The Coverage Percentage will be : 75.25%"

Images of some code snippets



```python
    def updateTotalCoverage(self, curr_metric, total_metric):
        # Compute the total_metric coverage and return it (list)
        # this changes if new coverage is seen for a
        # given input.

        if curr_metric not in total_metric and self.compareCoverage(curr_metric, total_metric):
            total_metric.append(curr_metric)
        return total_metric

class CustomMutator(MutatorBase):
    def __init__(self):
        pass

    # TODO : Implement this
    def mutate(self, input_data, coverageInfo, irList):
        # Mutate the input data and return it
        # coverageInfo is of type CoverageMetricBase
        # Don't mutate coverageInfo
```

```
[fuzz] Program took too long to execute. Terminated
[fuzz] Coverge for execution : [0, 5, 13, 14, 16, 17, 18, 19, 20, 21, 22, 26, 27]
The Coverage Percentage will be : 35.14%
[fuzz] Time Exhausted : 30.1880202293396
[fuzz] Terminating Fuzzer Loop.
Coverage : [[0, 5, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 37], [0, 5, 13, 14, 16, 18, 19, 20, 2
1, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 37], [0, 5, 13, 14, 16, 18, 19, 20, 21, 22, 23, 24, 25, 28, 31, 32, 33, 34, 37], [0,
5, 13, 14, 16, 17, 18, 19, 20, 21, 22, 26, 27, 28, 29, 30, 32, 33, 34, 37], [0, 32, 33, 34, 5, 37, 13, 14, 16, 18, 19, 20, 21, 22, 2
6, 27, 28, 31], [0, 5, 13, 14, 16, 18, 19, 20, 21, 22, 26, 27, 28, 29, 30, 32, 33, 34, 37], [0, 5, 13, 14, 16, 17, 18, 19, 20, 21, 2
2, 26, 27, 28, 31, 32, 33, 34, 37], [0, 5, 13, 14, 16, 17, 18, 19, 20, 21, 22, 26, 27]],
Corpus:
Input 0 : {':x': 50, ':y': 150, ':m': 50}
Input 1 : {':x': 50, ':y': -13, ':m': 50}
```