Program Analysis Project Report

Mahdi Jafarkhani
Matriculation Number: 3679412

January 31, 2024

1 Introduction

In this project, I implemented a dynamic analysis program using dyna-pyt library for Python language to analyze and slice a given file, given a single criterion line, and create another file with only those lines that are relevant to the slicing criterion. The full project is available in the zip file attached to this report, and also available throw the private GitHub repository. In this report, I discuss the approach taken, the results obtained, and any further developments that are needed to improve the project.

2 Approach

To achieve the requirements specified in the project description, I used some of the dyna-pyt hooks to help me throw analyzing the program. The full list of hooks and the main reason for their usage are available in Table 1.

To summarize the approach taken, I used 3 main dictionaries to collect information during the program (line by line) execution, namely `lines_info`, `variables_info` and `control_flow_stack`.

The first one, `lines_info`, collects information regarding each line of code, with each key being the line number, and its value is another data structure that captures which other lines are dependencies to this line.

The second important variable is `variables_info`. Each variable is a key to this dictionary and another data structure, `VariableMetaData`, as its value, captures meta-data about that variable, for example, which lines is its active (live) definition, what is its type, what are the elements (if array) or attributes information regarding that variable.

The last variable that is worth mentioning is a stack called `control_flow_stack`. This stack captures each control flow IID entering (the exact IID that is assigned during the instrumentation with dyna-pyt), and when exiting that control flow, removes its IID (and also all IID's that are above it in the stack). This approach is useful during control-flow analysis.

During code execution, each hook captures the information depending on which hook is entered and completes the information in the corresponding variables. At the end of the execution, I compute the slice in a recursive method, and after finding which lines should be kept, I create the sliced file in the same directory.

3 Challenges and Results

The first challenge that I faced during the project implementation was familiarising myself with the dyna-pyt library, which was not so challenging with the tutorials that were available in the Github repository.

During the implementation, with each test case that failed the approach, I needed to improve or change my code to adjust to the test case. For example, during the milestone 2 presentations, I found out that Python language is somehow call-by-reference for mutable objects, which made me change my approach for cases like we have in code snippet 3. Here, changing `p2.name` at line 9 triggers a change on object `p1`, which means that our slicing criterion at line 10 is also dependent on line 9. This test case, for example, made me add a new property to my `VariableMetaData` data structure, adding a list of references (or copies) of that variable, and also tracking their change along the program execution.

Another challenge was that it was necessary to restrict our analysis to the only part that is inside the criterion function. To accomplish this, I had to first find the slicing range (start line and end line of the slicing function), and inside each hook, I had to check whether it was in the range of slicing or not.

The results on the given test cases, and additional test cases that were added, are promising. There is no failed test case with the code that is available in this report. Running time is smooth and fast, and I did not encounter any critical issues like endless loops or any raised errors that were added to the code to secure the algorithm's functionality.

```
1  class Person:
2      def __init__(self, name):
3          self.name = name
4
5  def slice_me():
6      p1 = Person('Nobody')
7      p2 = p1
8      indefinite_pronouns = ['Everybody', 'Somebody', 'Nobody', 'Anybody']
9      p2.name = indefinite_pronouns[1]
10     return p1 # slicing criterion
11
12 slice_me()
```

Listing 1: A test case for call-by-reference for mutable objects

4 Further Works

This project is open to any improvement, including extending its slicing functionality from Intra-procedural to extra-procedural analysis, as is the case for most Python codes. Improvements in code clarity, which include reducing functions' length and breaking them into multiple methods are also applicable.

The biggest improvement that is worth noting refers to those parts of the code that analyze the AST with `if-then-else` or `isinstance` checks. These codes could be improved using a more generalized approach, and could widen the scope and limitations that are described in the project definition.

Dyna-pyt Hook	**Usage**
`read`	I use this hook when reading a variable, whether it is on the left-hand side or the right-hand side of a statement
`write`	This hook is used when writing a variable on the left-hand side, Also it is useful for checking whether an assignment should be considered as call-by-value or call-by-reference
`augmented_assignment`	This hook is useful for augmented assignments like `p.age += 1`. A limited approach with `writing` is also applicable to this hook.
`read_subscript`	This hook is useful for subscript reads like `a[10] = 0`. A limited approach with `writing` is also applicable to this hook.
`function_enter`	This hook is useful for finding what is the range of our analysis. As stated in the project description, a function named `slice_me`, is considered as our intra-procedural analysis and other hooks are aware that whether we entered this function or not.
`end_execution`	This hook is useful as we are confident that the execution is complete and we are ready to compute the slice and create the sliced file.
`enter_if`	This hook triggers an enter to an `if` control flow, which means that I add its IID into the stack, together with its line number.
`exit_if`	This hook triggers an exit from an `if` control flow, which means that I remove its IID from the stack, together with all IID's that are added after entering this flow.
`enter_for`	This hook triggers an enter to a `for` control flow, which means that I add its IID into the stack, together with its line number.
`exit_for`	This hook triggers an exit from a `for` control flow, which means that I remove its IID from the stack, together with all IID's that are added after entering this flow.
`enter_while`	This hook triggers an enter to a `while` control flow, which means that I add its IID into the stack, together with its line number.
`exit_while`	This hook triggers an exit from a `while` control flow, which means that I remove its IID from the stack, together with all IID's that are added after entering this flow.

Table 1: Summary of dyna-pyt hooks used in the project