CS639A: Program Analysis, Verification And Testing
Users Online : 14

Assignment #2: Constructing Control Flow Graphs

Submitted on 1/9/2021 22:12

Instructions

- You are given an extra 10 minutes after due time to submit your assignment.
- However, please note that any submissions made after the due time are marked as late submissions.

Assignment #2: Constructing Control Flow Graphs

Question:

Assignment 2: Constructing Control Flow Graphs

Disclaimer: The assignments in this course will be based on a new (fun) program analysis framework that we are developing. We will require you to submit responses to short survey questionnaires that will allow us to improve ou framework and also publish this work as a research paper. Though we will require you to reveal your identity with your responses for the purpose of accounting, your feedback will be completely anonymized for the purpose of the publication. Also, your responses will have no bearing on your grades, so we will request you to honestly answer th questionnaire; in fact, we will value critical feedback as it will help us improve our framework. The estimated time t fill one questionnaire will be less than 10 minutes, and we will not have more than 10 questionnaires across the whole semester. If you are fine with it, you will need to sign a release that allows us to publish your anonymized responses in a research paper.

If you are not willing to be part of this, please send me a mail mentioning the same. In that case, we will provide alternate assignments (somewhat similar ones in spirit) on an existing compiler framework (like LLVM) instead of ou framework. Please realize that using such frameworks will be much more difficult (as they are full-fledged compiler of large languages), less fun (our framework is for a small fun language, Turtle), and may limit discussions (hoping most students are fine with being part of this). Lack of a simple and diverse framework to learn program analysis is our motivation for building this. We hope that you will help us with our endeavor.

This assignment is to be implemented in **one** person team.

We will now build a program analysis tool for the Turtle language. For this purpose, you will use our in-house framewor (currently being developed by your TAs). The current version is attached. You may report bugs at: https://github.com/CS639A-PAVT/BugTracker

Currently, you can run the program by proving a Turtle file; the program will dump the intermediate representation in human-readable form. Please refer to **README.md** file in the project to install some dependencies.

Objective

You will complete the following two procedures in the file "submission.py" that is automatically invoked by our framework.

CS639A: Program Analysis, Verification And Testing
Users Online : 14

2) dumpCFG(cfg): It consumes the CFG constructed in the previous step and generate a dot file that can be viewed by the graphviz utilities (dotty).

Your implementation should be clear, well-commented and modular, as you will extend this tool in the subsequent assignments. You may add additional procedures, classes and files as you may require.

Input

A Turtle program with only the following statements:

Assignment statement - *e.g.* :varA = 10

If statement - if <condition> [<statement list>]

Repeat statement - repeat <number> [<statement list>]

Move command - *e.g.* forward 20 , back 10, up 4, down 5

Pen command - penup *or* pendown

You can find a few examples in the exmaple directory of the repository provided to you.

Format for IR is as follows

IR is represented in the form of list of tuples as shown below.

<sequence number> <condition stmt> [<relative seq number>]

<sequence number> <other than condition stmt> [<relative seq number>]

A tuple consists of sequence number -- index of a tuple in the list--, assignment/condition stmt followed by relative seq number. Program counter (pc) points to the index of IR list and is initialized to zero.

In the case of condition stmt if evaluation of the condition is "True" pc is incremented by "1" otherwise, pc is modified b adding relative seq number. For all other stmts, pc is modified after execution of each stmt by adding relative seq numb to pc.

Output

Dump the CFG as a dot file and plot it. You can use GraphViz to render the dot file into an image (Note that, using GraphViz to plot dot files is not mandatory; you can use any tool you want). There is a python package available for GraphViz here .

Example

Input:

Turtle program

CS639A: Program Analysis, Verification And Testing
Users Online : 14

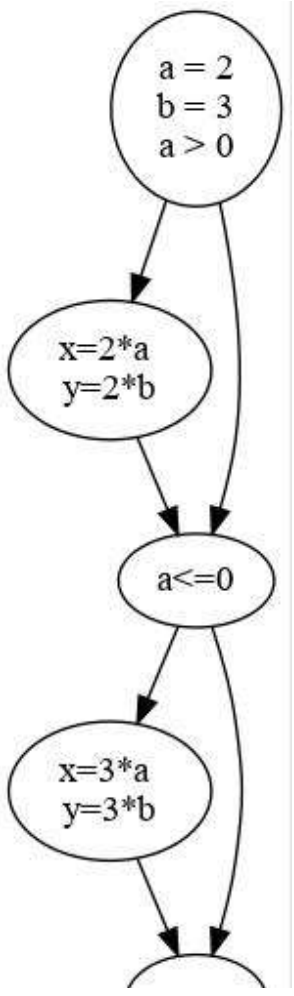
```
  :x = 2 * :a
  :y = 2 * :b
]

if :a <= 0 [
  :x = 2 * :a
  :y = 2 * :b
]

========== IR ==========

0 :a = 2 [ 1 ]
1 :b = 3 [ 1 ]
2 :a > 0 [ 3 ]
3 :x = 2 * :a [ 1 ]
4 :y = 2 * :b [ 1 ]
5 :a < 0 [ 3 ]
6 :x = 2 * :a [ 1 ]
7 :y = 2 * :b [ 1 ]
```

Output:



CS639A: Program Analysis, Verification And Testing
Users Online : 14

Deliverables

The source code **of only your implementation**: submission.py and any other files you may write.
A set of test cases (at least 5) with expected output. Provide the CFG as an image for each test case.
A **Readme** file (txt/pdf/doc/docx) which will give a short report (half to one page maximum) on how you accomplished this task and how to build and run your source on any input python file.
You should NOT package the whole framework.

The quality of all the above would affect your marks.

Submission Format

Your submisssion **MUST** be in the following format:

The submission should be a **zip** file.
The zip file should be named as **assignment_"number"_"Roll-of-student"**.
The zip file should contain (1) a directory named **source** which will contain the source code, (2) a directory named **testcases** which will contain the test cases written by you and the corresponding outputs in separate text files, (3) a **Readme** file (txt/pdf/doc/docx).

Please note that your submission will **NOT** be graded if you do not follow the format. Furthermore, we will use the **Readme** file provided by you to build and run your code. Therefore, please make sure that the Readme is clear. We cannot grade your submission if we cannot run it on our system.

Some important comments

Before doing anything "extra" (which might fetch bonus marks), first complete the basic expectations from your implementation.
Program analysis tools are expected to display their results in an user-friendly manner; a user would never like to use a tool that simply spits out a bunch of numbers. So, display the results from your tool suitably.

kachua_3.zip

Uploaded Files:

assignment_2_21111037.zip

Grades:

Marks: 100