Control Flow Graph Generator in Python

Steven Qin
20819760
s45qin@uwaterloo.ca

Huijie Chu
12345678
h25chu@uwaterloo.ca

Tiankai Jiang
20834939
t57jiang@uwaterloo.ca

May 26, 2020

Abstract

A control flow graph(CFG) is a graphical representation of the flow of the computation during the program execution. Since it represents the flow of the logic, it can be used for static analysis or compiler optimization, etc. In this project, we aim to build a static control flow graph generator in Python from scratch using abstract syntax tree and graphviz.

1 Problem Statement

Basic block is a series of program instructions that has only one entry point and one exit point. And control flow graph is a directed graph in which the vertices represent the basic blocks and the edges represent control flow paths. Using CFG, the control flow of the program can be shown easily, making it easier for developers to check out logic error or find out the paths that are not covered by the current tests. Thus, a neat and efficient control flow graph generator is required for software developent purpose. In this project, a control flow graph generator will be built in Python, with initial support for source code in Python and hopefully with other languages support later. The generator is static, which means it will generate the graph without executing the input source code, therefore the generator should be highly efficient.

2 Approach

There are mainly two steps for CFG generation. First find the basic blocks of the source code and then transform them into graph representation. To find the basic blocks of the code, simply find the begin and the end of a block. The begin of a basic block includes the first instruction of the code, the next line of the conditional instruction and the function call. And the end of the block could be the conditional instruction, the return, throw or exit keyword and the last instruction of the code. Since a module called Abstract Syntax Trees(AST) is provided in Python, we can directly transform the parsed result of AST to CFG. However, the technique mentioned above is still useful if we want to generate CFG from other languages like C++.

An abstract syntax tree is a tree representation of the abstract syntactic structure of the source code, with each tree node represents a keyword, a variable or an operator of the code. It contains the basic block information. Thus, to generate CFG from Python code, we can start from its AST directly.

The output format of the graph could be a dot file or an image generated by that dot file using package graphviz in Python.

3 Framework and Toolchain

Since we build this CFG generator from scratch, there is no framework or toolchain required. We plan to use AST package in Python for syntax parsing and graphviz package for graph generation.

4 Benchmarks

Our own LeetCode solutions(about 200 of them, each contains about 10-40 LOC) in Python will be used to test out program. The solutions have a moderate length and contains lots of conditional instructions, which are perfect for testing purpose. To test the runtime performance of the program, our past Python projects($>$ 200 LOC) will also be used, providing a benchmark for larger source codes and it will reflect a performance closer to the daily application scenarios.

5 Plan of Action and Feasibility Analysis

Week	Action
3-4	Read papers & tutorials about CFG and Python AST and algorithm design.
5-8	Implementation of the control flow graph generator.
9-10	Testing
11-12	Add support to another language if possible.
13	Demo

6 Demo

The program is expected to recognize the correct control flow of the input source code and generate the correct CFG. The control flow is not restricted to keywords such as if, else, while in normal syntax, but also in special syntax like list comprehension or lambda expression. And it shuold detect syntax errors in the source code.