COM S 413/513 Project 4: Analyzing software changes and versions

Learning Objectives

1. Gain experiences with LLVM and MVICFG, the C/C++ program analysis tools
2. Understand the challenges of analyzing program changes and versions
3. Develop extensions and contribute to existing program analysis tools

Description

In this homework, you are going to work with one of the prominent program analysis platforms llvm https://llvm.org/ and one of the frontier static analysis tools, called multi-version control flow graphs MVICFG https://dl.acm.org/citation.cfm?id=2568304. Here are the set of instructions that can guide you through the work.

1. Create a gitlab account and get access to the MVICFG repo.
2. Follow the manual of the MVICFG tool, install the docker and get the MVICFG running with the *test* program.
3. (20 pt) Extending the MVICFG framework and write two functions path_added and paths_removed to report newly added acyclic paths and newly removed acyclic paths for a given code churn.
4. (3 pt) Running the test case (3 versions) given in the MVICFG repo to demonstrate the output of your code.
5. (5 pt) Creating and running your own test case (3 versions) to demonstrate the output of your code.
6. (10 pt) Finding and testing 3 versions of a real-word program to demonstrate the output of your code.
7. (2 pt) Write down the rationale of test case design and benchmark selection.
8. (10 pt) Write a summary report that includes the following:
    a. (6 pt) A table that summarizes the three tests you have run.
    b. (4 pt) Can MVICFG and your implementation help develop and assure software changes? If so, how?

Version pairs	Code churn size	MVICFG size (nodes, edges)	*Acyclic paths added	*Acyclic paths removed	time used
Test: v1-v2					
Test v2-v3					
Your test program v1-v2					
Your test program v2-v3					
Real-world program v1-v2					
Real-world program v2-v3					

* if loops are involved, you can iterate the loop once to ensure all the statements in the loop are covered. If the loop contains a if-then-else branch, you will generate two paths, one covers the false branch and one covers the true branch.

Deliverable and Grading Criteria (50 pt)

Please zip the following files and submit the zipped file to canvas under the “project 4: analyzing changes and versions” column.

(1) Step 3 (20 pt): source code of the two functions you implemented. We will use our test cases to test your code. If the output is correct, you get the full points. If the output is partially correct or incorrect, we will inspect your code to provide partial credit.

(2) Steps 4--6 (20 pt): please submit all the source code and intermediate results
   (a) (3 pt) Run successfully with the given test case (show screenshot)
   (b) (2 pt) A correct test case created by yourself (provide source code of three versions)
   (c) (3 pt) Run successfully with your test case (show screenshot)
   (d) (4 pt) Find a real-world program, consider programs with big sizes/multiple versions, and used by people (provide source code of 3 versions)
   (e) (6 pt) Make the real-world program working with MVICFG and show output

(f) (2 pt) Elaborating the rationale used for creating your own test cases and for finding the real-world program. You can add this to readme.txt

(3) Step 5 (10 pt): a summary report.

The homework is due April 10, Fri 11:59pm. You can continuously submit your homework without a penalty after the deadline. But once I started grading the homework, I am no longer accepting late homework or modifications. Start early!