CSC410

Homework 3

Due on Monday October 30, 2017 before 11am

Submit on Markus.

About homework 3

Homework 3 and Homework 4 are the two parts of a homework on the topic of Data Flow Analyses. This is the first part which is a practical programming exercise developing an example data flow analysis with a tool. The second part (i.e. homework 4) will be a handwritten assignment on the theory behind data flow analyses.

Note: homework 3 is basically a freebie. The design is almost trivial, and a framework is given to you to turn your design into an implementation with very little effort. We expect an average-good student to do this in about 2 hours. So, a group of 4 students should have no problem to do this in virtually no time. What will you do with all the extra times on your hands? Start doing your course projects!

Problem 1

Upwards Exposed Uses. Upwards exposed uses serves a similar purpose to reaching definitions analysis. Instead of finding which definitions reach each point, we find which *uses* of a variable have not yet been matched with one or more definitions. Given a control flow graph $\langle V, E\rangle$ the use of the variable x at vertex v is upwards exposed at a vertex u (i.e., $\langle x, v\rangle \in UEU(u)$) if

- x is read at v, and
- there exists a path from u to v along which no vertex assigns to x.

(a) (10 points) Formally define the property space for this problem (i.e. a set and its corresponding meet operator).

(b) (20 points) Define the (statement) transformer functions for this analysis (stick to the simple language that is used on the slides for all the other example analyses that you have seen).

(c) (10 points) Fully define the analysis by characterizing whether it is forward or backward, what the initial nodes are, and what value is given to these initial nodes and all the other nodes before the default iterative work-list algorithm starts.

(d) (60 points) Having done all of the above properly, implement your design in SOOT and get an analyzer for the *upward exposed uses analysis*.

Assignment Format and Guidelines on Submission

For this assignment, you will be implementing data flow analyses using SOOT, which was introduced to you during the last tutorial.

For the problem you are given the base Java code. You will modify the file `UpwardExposedUses.java`. You will **not edit any other file**. You are expected to submit **only the file** `UpwardExposedUses.java`. (Any helper or additional classes you may need to create should be created within this file itself).

You must create your `UpwardExposedUses` class by extending any of the Data Flow Analysis classes provided by Soot, and overriding the appropriate functions accordingly. You must not use any library or package (other than Soot) or code from any such library or package- all code you submit must be your own.

Input-Output Format

The input will be in the form of a Java .class file. For the output you should **print all the upward exposed uses to a file** called `exposed-uses.txt` in the same location from where the code is run.

Note that you **should not** print the upward exposed uses to STDOUT. Any output to STDOUT or STDERR will be ignored. If the program creates any .jimple/.jimp files in the directory *sootOutput*, let them stay there; you should not delete them.

Printing upward exposed uses

You should print, to the file `exposed-uses.txt`, all upward exposed uses with each upward exposed use taking exactly 3 lines. If the the use of the variable x at node v is upwards exposed at the exit of node u (i.e. if $\langle x, v\rangle \in UEU(u)$) then you would print on the first line the node u, on the second line the node v, and on the third line the variable x. Make sure you have the order correct. You should make sure that each node fits within a single line. If any node takes up more than one line, remove all newline characters from the output of the node.

```
z = 2
y = 10
x = y*z
```

For example in the case of the above code, there are just three nodes in the graph- `z = 2` `y = 10`, and `x = y*z`, in which case the output to the file `exposed-uses.txt` should be:

```
z = 2
x = y*z
z
y = 10
x = y*z
z
y = 10
x = y*z
y
```

While actually running your code for the above case the names of the nodes, branch labels, and variables in the control flow graph passed to your analysis class may vary slightly depending on Soot's pre-processing, and so the names of the nodes and variables printed to the file may be different; that is okay.

You may print the exposed uses in any order. But you should cover all upward exposed uses for the exit points of all nodes and you should not repeat the same exposed use twice in the file. You are given a particular test case and its expected output in the base code.

Evaluation

We will test your answer on CDF/CSLab machines by placing your submitted `UpwardExposedUses.java` file in the base code source directory, compiling it, and running it on examples. The code should compile and run as in the `test.sh` script. You should make sure your `UpwardExposedUses.java` file compiles and runs with the provided base code on CDF/CSLab machines.

The code will be tested on different examples. For each correct output, you will get an equal fraction of the problem points. In other words, if your code returns the correct output on all tests, you will get a full mark, and for every wrong answer you will lose (proportionally) some points. Beyond this, there are no partial marks for any partial efforts.

If the code fails to compile, returns a runtime error due to your fault, or fails to print the results in the correct format to the `exposed-uses.txt` file you may not get any marks.

Summary of submission files

- A PDF file for parts (a-c). Call it `hw3.pdf`.
- The file `UpwardExposedUses.java` as described above for part (d).