Assignment 2 (Programming): Dataflow Analysis

15-819O: Program Analysis
Jonathan Aldrich
jonathan.aldrich@cs.cmu.edu

Due: Monday, February 4, 2013 (11:59 pm)

80 points total

Assignment Objectives:

- Implement a dataflow analysis in a code framework built based on the concepts of flow functions and lattices.

Handin Instructions. Turn in a zip file electronically via Blackboard for Assignment 2. The zip file should contain an Eclipse project with your analysis implementation. At the top level in the project should be output.xxx - a screenshot in some common graphics format.

1 Sign Analysis Implementation

In this assignment, you will implement your sign analysis for the Java programming language using Soot's dataflow analysis capabilities. You may choose to use some other dataflow analysis engine and/or some other language; if you want to do so, contact the instructor to discuss whether any aspects of the assignment need to be adapted.

In Java, integer variables are separate from variables that hold references, booleans, floating point values, etc. Your implementation need only track information for variables of type int. Your analysis only needs to track the sign of local variables. Any use of fields, arrays, method parameters or results can be considered to have unknown sign.

You should implement your analysis by defining a class to represent your lattice. Your dataflow analysis will need to define the operations for your lattice, as well as the flow functions. Don't forget to ensure that your lattice handles equals() and hashCode() correctly!

In order to drive the flow analysis, use a BodyTransformer that creates a flow analysis for each method body. Inside the BodyTransformer, use tags to report the values of variables used as array indexes–e.g. whether they are definitely negative (an error) or possibly negative (conceptually a warning).

Your analysis should cover variable copies, integer constants, addition, subtraction, and multiplication as precisely as possible given your lattice. You are not required to correctly analyze other operations, though your analysis should not crash.

Question 1 (20 points).

> Run your analysis on TestSign.java. Turn in a screenshot showing the tags generated for TestSign. When capturing the screenshot, resize the window if necessary to show all the code. Do **not** change TestSign.java.

Question 2 (60 points).

> Turn in your analysis code. Your code should follow the basic design described above.
>
> We reserve the right to run your analysis code on examples other than TestSign.java, looking both for accuracy of the analysis and its robustness (i.e. it should not throw unexpected exceptions when run on a larger codebase).