Assignment 5 (Written/Programming): Dataflow Analysis

17-654/17-754: Analysis of Software Artifacts
Jonathan Aldrich (jonathan.aldrich@cs.cmu.edu)

Due: Thursday, March 2, 2006 (5:00 pm)

100 points total

Turn in a file named <username>-17654-A5.{zip}, where username is your Andrew id. The zip file should contain the file answers.xxx (the answers to text questions in txt, pdf, or Word (doc) format), each of the .java files you wrote, and output.xxx (either analysis output in .txt format or a screenshot in some common graphics format). At the top of answers.txt, state your name, Andrew id, and how long you spent on the assignment.

Assignment Objectives:

- Precisely define an analysis using lattices, abstraction functions, and flow functions.
- Implement a dataflow analysis in a code framework built based on the concepts of flow functions and lattices.
- Demonstrate understanding of local soundness by providing a counterexample to local soundness for a buggy analysis specification.

1 Valid Pointer Analysis Specification (30 points)

Your first task is to precisely define a valid pointer analysis for WHILE. A valid pointer is a pointer which it is safe to dereference (i.e. it is not null and does not point to garbage). Assume that WHILE has been extended with the following syntactic construct:

$$\begin{array}{rcl} a & ::= & \ldots \\ & | & \&x \end{array}$$

The $\&x$ expression takes the address of some variable x, resulting in a valid pointer. For the present, we will assume WHILE has a C-like pointer model, where regular integer variables can hold pointer values. We represent null with the constant 0. We also assume that integer constants–including zero–are not valid pointers.

Question 1.1 (15 points).

> Design a lattice for a single variable. Your lattice should be able to represent both definitely valid and definitely not valid states, as well as possibly valid. Define the lattice by giving (a) the set of lattice elements and (b) the ordering relation between them, (c) the top element and (d) the bottom element.

Question 1.2 (5 points).

> What is the initial analysis information before the first statement of each function? Justify your choice (more than one answer may be correct, so long as it is justified).

Question 1.3 (10 points).

> Define the flow functions for your analysis, using the notation given in class. Naturally, you will need to include flow functions for the new expression form $\&x$, as well as for integer constants such as 0 (which are invalid pointers).

2 Valid Pointer Analysis Implementation (50 points)

Next, you will implement your valid pointer analysis for the Java programming language using Crystal. In Java, integer variables are separate from pointer variables, so your implementation need not keep track of whether integer variables and expressions are valid or not (you may if you wish; if so, these are all invalid). However, pointer variables are either valid or null (with null taking the place of 0 in WHILE). Thus, your analysis should track the nullness of local variables, but should assume that values in fields and method parameters could be either null or non-null.

You should implement your analysis by defining a new subclass of LatticeElement which describes a valid pointer lattice for a single variable, and using this with the TupleLattice class to build a tuple lattice for vlaid pointer analysis. You should subclass FlowAnalysisDefinition to define your flow functions as well as the initial and bottom analysis info.

Finally, you should subclass AbstractCrystalMethodAnalysis to write a client analysis that will report invalid pointer dereferences. Your analysis should produce a warning in the errors window or the console for each of the errors marked with comments in the test file TestNull.java.

To make the assignment more feasible to grade, all the classes you write must be put in the package edu.cmu.cs.crystal2.asst5. The client analysis class (the one that inherits from AbstractCrystalMethodAnalysis and needs to be registered in CrystalPlugin) must be named NullPointerAnalysis.java.

Question 2.1 (10 points).

> Turn in a screenshot of the problems window, or the text produced by your analysis if you wrote to the errors window. When capturing the screenshot, resize the window if necessary to show all the errors.

Question 2.2 (40 points).

> Turn in your analysis code. Your code should follow the basic design described above. Remember to use package edu.cmu.cs.crystal2.asst5 and client class name NullPointerAnalysis.java.
>
> **Important note:** Your code must be robust. For example, it should not throw unexpected exceptions when analyzing valid

Java code (these exceptions will show up on the Crystal console). We will be running your code on a large codebase to check its robustness, and we recommend you do so as well. One simple approach is to run your analysis on the Crystal codebase.

3 Local Soundness (20 points)

Consider the following hypothetical flow function for constant propagation:

$$f_{\text{CP}}(\sigma, [[...]_n + [...]_m]_k) = [t_k \mapsto (\sigma(t_n) -_\top \sigma(t_m))]\sigma$$

Here the operator $-_\top$ is like $op_\top$ defined in lecture: the same as $-$ for integers, but yeilding $\top$ if either of its operands is $\top$.

This flow function is "obviously" incorrect. However, to prove that it is incorrect, you must show that it violates the criterion of local soundness defined in class. That is, you must find an environment η and a statement S such that $(\eta, S) \mapsto (\eta', S')$ and $\alpha_{CP}(\eta') \not\sqsubseteq f^*_{CP}(\sigma, S)$ with $\sigma = \alpha_{CP}(\eta)$.

Question 3.1 (5 points).

Find a pair (η, S) that illustrates the local unsoundness. What are η and S?

Question 3.4 (2 points).

What is $\sigma = \alpha_{CP}(\eta)$?

Question 3.3 (3 points).

Assume $(\eta, S) \mapsto (\eta', \mathit{skip})$. What is η'?

Question 3.5 (2 points).

What is $\alpha_{CP}(\eta')$?

Question 3.6 (4 points).

What is $\sigma' = f^*_{CP}(\sigma, S)$?

Question 3.7 (4 points).

Show that $\alpha_{CP}(\eta') \not\sqsubseteq \sigma'$