210: Compiler Design

Fall 2015

Homework 5

Due date: Dec 18, 2015, 23:59

15 points

1 Introduction

This homework is the last one. It makes your compiler more useful and paves the way for the advanced compiler course in the next semester. Your task is to implement two data-flow analyses – one that reports a likely mistake to the programmer, and one that gives the compiler more information to optimize your program.

2 Task Description

Your task is to implement two analyses that compute *intra*-procedural dataflow information:

- Finding uses of uninitialized variables (*-uninit*): Generate a semantic error (i.e., `POSSIBLY_UNINITIALIZED`) whenever a (potentially) uninitialized local variable is used as an operand.
- Busy expressions (*-bexpr*): Identify busy expressions and print a message in the console that shows useful information about these expressions (e.g., expression line number with the expression string or AST sub-tree).

These analyses are separate and independent of each other, i.e. one analysis does not depend on the analysis performed by the other.

You *must* provide two separate flags (*-uninit* and *-bexpr*) for your compiler to enable/disable the analyses, so that we can test your analyses separately if needed.

You need to test each analysis in isolation and you are free to add other flags that help you in testing your implementation.

For this homework `*we do not provide*` a reference implementation that runs these analyses. As usual, you are required to provide additional test-cases to show how your analysis works and which cases are handled/not handled.

2.1 Simplifications

Your global dataflow analyzer is not required to work on *arrays* and *fields*. If a method calls other methods, your dataflow analyzer must still be correct and compute a conservative approximation that is as accurate as possible. You do not have to implement an *inter*-procedural analysis.

3 Deadline extensions

The university's rules prohibit setting the due date of an assignment to a date that is after the last day of classes. *Please plan ahead!* If for some unexpected reason you feel that an extension is necessary, please contact us.

4 Working on the solution

Please make sure that your code is committed into the `HW5` directory of your team's repository by the due date indicated at the beginning of this document. You will be working with your implementation of `Homework 4` please make sure that you **copy** your `HW4` solution into the `HW5` directory before you start working on `Homework 5`. In addition make sure that your final submission works correctly in the absence of these analyses, i.e. when the flags above do *not* turn on the analyses.

Finally please put a file named `README_ANALYSES` into the `HW5` directory. In the file you should provide a short description of the analyses you have implemented, and also a description of the cases handled/not handled by your code.