CMSC 430 (Spring 2009)
Practice Problems 5

1. Optimizations

   (a) How can compiler transformations improve a program?

   (b) What does the compiler need to consider when applying optimizations?

   (c) What are the different scopes of compiler optimizations? What are the tradeoffs when considering what scope of optimizations to use?

2. Local optimizations

   Consider the following code.

```
(1) a := 1
(2) b := f + a
(3) c := a
(4) d := f + a
(5) e := f + c
(6) f := b
(7) g := f + a
```

   (a) Build a DAG for the code.

3. Control flow analysis

   For the following problems, consider this code:

```
<S1>            a := 1
<S2>            b := 2
<S3>     L1:    c := a + b
<S4>            d := c - a
<S5>            if (...) goto L3
<S6>     L2:    d := b * d
<S7>            if (...) goto L3
<S8>            d := a + b
<S9>            e := e + 1
<S10>           goto L2
<S11>    L3:    b := a + b
<S12>           e := c - a
<S13>           if (...) goto L1
<S14>           a := b * d
<S15>           b := a - d
```

   (a) What are the basic blocks?

   (b) What is the control flow graph?

   (c) Depth-first order selects nodes in the order they are visited (start by visiting the root node) and then recursively visiting every child of each node (if the child has not been visited before). Note that the order in which children are visited is random.

   What are all the possible results of depth-first traversal on the control flow graph?

   (d) Using depth-first order, is it possible to visit a child before its parent? For which depth-first ordering(s) of the control flow graph does this occur?

   (e) Postorder selects nodes (starting from root) *after* visiting every child of that node (if the child has not been visited before). Note that the order in which children are visited is random.

   What are all the possible results of Postorder traversal for the control flow graph?

   (f) Reverse Postorder simply reverses the node ordering found by a Postorder traversal of the graph. What are the possible Reverse Postorder traversals of the control flow graph?

   (g) Using Reverse Postorder, is it possible to visit a child before its parent? Why or why not?

4. Reaching definitions

   Reaching definitions (RD) for a point in the program p is defined as the set of definitions of a variable for which there is some path from the definition to p with no other definition of that variable. Calculate reaching definitions for the code in the control-flow graph problem.

   (a) What is the dataflow equation for RD?

   (b) In what direction is RD calculated? I.e., does information flow forwards or backwards in the CFG?

   (c) Calculate GEN, KILL for each basic block.

   (d) What is a good initial value for RD for each basic block?

   (e) Solve the data-flow equations in reverse Postorder. Show your work.

5. Live variables

   Live variables (LV) for a point in the program p is defined as the set of variables x for which there is some path from p to a use of x with no definition to x on the path. Calculate live variables for the code in the control-flow graph problem.

   (a) We define LV(b) for a basic block b to be the set of live variables at the *end* of b. What is the dataflow equation for LV?

   (b) In what direction is LV calculated? I.e., does information flow forwards or backwards in the CFG?

   (c) Show GEN, KILL for each basic block.

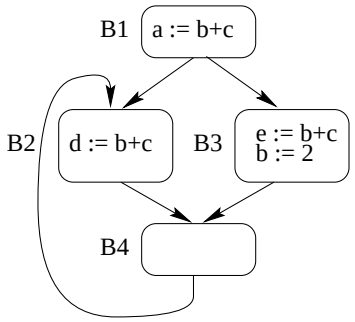
   (d) What is a good initial value for LV for each basic block?

   (e) Solve the data-flow equations in rPostorder. Show your work.

6. Available expressions

Available expressions is a data-flow analysis problem whose solution is used to guide global common subexpression. It calculates AVAIL, the expressions available at the beginning of each basic block.

Consider the following code. Assume that `b+c` is the only expression of interest:



(a) What is the data-flow equation for AVAIL?

(b) Give GEN and KILL (needed by AVAIL) for each basic block.

(c) What is a good initial value for AVAIL for each basic block?

(d) Calculate AVAIL. Show all steps, including values for AVAIL and the order basic blocks are analyzed.

---

7. Data-flow lattices

Prove the following properties of lattices:

(a) Show that $a \leq b$ and $b \leq c$ implies $a \leq c$

(b) Show that $a \leq (b \wedge c)$ implies $a \leq b$

---

8. Data-flow frameworks

(a) When estimating each of the following sets, tell whether too-large or too-small estimates are conservative. Explain your answer in terms of the intended use of information.

i. Available expressions

ii. Reaching definitions

iii. Live variables

(b) What properties are necessary to ensure an iterative data-flow analysis framework terminates?

(c) What properties are necessary to ensure an iterative data-flow analysis framework terminates with the meet-over-all-paths solution?