CS 243 Assignment 1

Data Flow Analysis

Due: January 23rd, 11:00 am

This is a written assignment, every student must hand in his or her homework. Bring your homework to class on Wednesday, January 23rd. SCPD students may submit their homework by e-mail via scpd-distribution@lists.stanford.edu or give your homework to the courier.

If your solution to any of the questions involves defining a data flow analysis, specify the algorithm fully by answering the following questions:

a. What is the direction of your data flow analysis?

b. What is the set of values in the semi-lattice?

c. Draw a diagram of the semi-lattice, identifying the top and bottom elements clearly.

d. How do you initialize the information at the entry/exit nodes?

e. How do you initialize the internal nodes?

f. Define the transfer function of a basic block. You may assume that each basic block contains only one statement.

g. Is your framework monotone? (no explanation necessary)

h. Is your framework distributive? (no explanation necessary)

i. Will your algorithm necessarily converge? Explain.

1. Is the following a meet operator? Please answer yes or no. No explanation is necessary.

   a. Maximum value (on integers)

   b. Product (on integers)

   c. Addition (on integers)

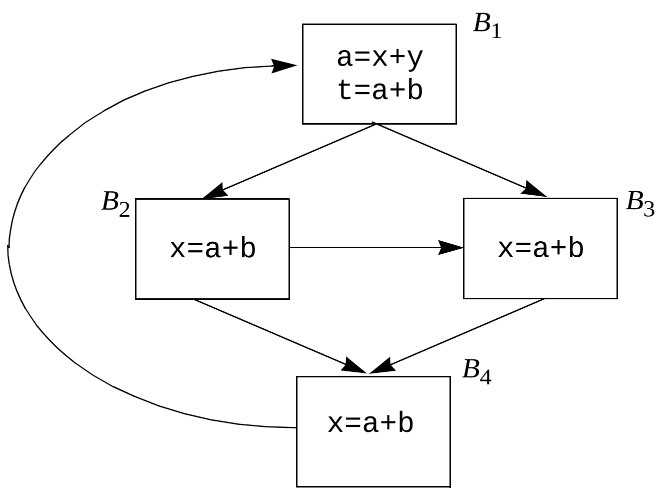
   d. Division (on integers)

   e. Set subtraction (on sets)

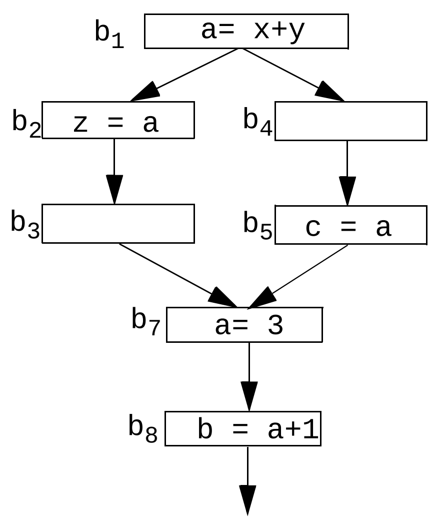
   f. Sum mod 2 (on the set {0, 1})

   g. Product mod 2 (on the set {0, 1})

2. Compute the available expressions (Chapter 9.2.6 in ALSU) on entry and exit for each basic block in the flow graph below.



3. We say that a program point *p* belongs to a live range of a definition *d* (`u=x+y`) iff the value assigned to `u` in definition *d* may be accessed via a use of variable `u` along some path in the flow graph starting at *p*. Consider for example the code fragment below:



The live range of the definition `a = x+y` is {exitpoint(b_1), entrypoint(b_2), entrypoint(b_4), exitpoint(b_4), entrypoint(b_5)}.

Similarly, the live range of the definition `a = 3` is {exitpoint(b_7), entrypoint(b_8)}.

This concept of live ranges can be used to increase flexibility in register assignment. For example, we can assign variable `a` in blocks {b_1, b_2, b_4, b_5} to register R1 and assign `a` in blocks {b_7, b_8} to register R2.

Describe an algorithm to find the live range of every definition in a program.

4. Suppose you have carefully defined a forward data flow algorithm for a framework that is distributive and has only finite descending chains.

   a. A colleague of yours accidentally deletes the code that initializes the entry node. Without telling you, he initializes the entry node to $\perp$ (bottom). Does that change the result of the analysis? Why?

   b. Suppose instead your colleague initializes all the *internal* nodes to $\perp$ (bottom).

      i. Will this algorithm give a safe answer for all flow graphs? (Recall that a data flow solution is safe if it is no bigger than the ideal solution).

      ii. Will this new algorithm give a safe answer for some flow graphs? If so, which ones?

      iii. Will this new algorithm give the meet-over-paths answer for all flow graphs?

      iv. Will this new algorithm give the meet-over-paths answer for some flow graphs? If so, which ones?

5. A propositional Boolean formula consists of Boolean variables connected by logical operators. Example operators include logical and ($\wedge$) logical or ($\vee$), and logical negation($\neg$). Let F be the set of propositional boolean formulas over the universe of atoms U. Consider the partial order $\leq$ over F induced by logical implication $\rightarrow$. Specifically, for any two formulae $f_1, f_2 \in F$, $f_1 \leq f_2$ if and only if $f_1 \rightarrow f_2$. ($f_1 \rightarrow f_2 \equiv \neg f_1 \vee f_2$). Give a semi-lattice that defines this partial order.