CS 701

Final Exam

Thursday, December 15, 2005

11:00 a.m. — 1:00 p.m.

3418 Engineering

Instructions

Answer question #1 and any three others. (If you answer more, only the first four will count.) Point values are as indicated. Please try to make your answers neat and coherent. Remember, if we can't read it, it's wrong. Partial credit will be given, so try to put something down for each question (a blank answer always gets 0 points!).

1. (1 point)
   In the context of data flow analysis, DFO abbreviates:
   (a) Departments of Fisheries and Oceans.
   (b) Denver Field Ornithologists.
   (c) Deutsche Forschungsgesellschaft für Oberflächenbehandlung.
   (d) Depth First Order.

2. (a) (20 points)
   A key step in putting a program in SSA (static single assignment) form is placement of the ϕ functions. Assume that all assignments to variables have been rewritten so that each variable is now assigned to exactly once. Give an algorithm to determine, for a set of variables $x_0, \ldots, x_n$ derived from a single variable x, where to place ϕ functions and the appropriate arguments for each ϕ. How are the arguments (the x_i values) of each ϕ determined?

   (b) (13 points)
   For a given definition of a variable, its **use set** is the set of all basic blocks that might use that definition. Show that once a program is in SSA form, computation of use sets is greatly simplified. In particular no analyses beyond those needed to put a program in SSA form are needed.

3. In class we studied both a work-list algorithm and an iterative DFO-driven algorithm for solving data flow problems.

   Consider a third (forward-flow) approach in which we first set In_0 to its defined initial value and all other In_i and Out_i values are set to top. We then place all basic blocks into a set. A block, b_i, is *randomly* selected, and Out_i is computed. Then a second different block, b_j, is selected and Out_j is computed. This process repeats until all blocks are selected once in some random order. If any Out_i value changes during this random visitation of blocks, the process is repeated (blocks are selected in random order and transfer functions are evaluated). We continue until no Out_i value changes during a full random traversal of basic blocks.

   (a) (11 points)
   Show that if we use a meet semilattice with finite height, and have monotone transfer functions, then our new data flow algorithm must terminate with a valid solution to the data flow problem.

   (b) (11 points)
   Show that the solution our new algorithm computes is the same at that computed by the DFO-driven algorithm we studied in class.

   (c) (11 points)
   Do you expect this new algorithm to be faster or slower (on average) than our DFO-driven algorithm? Why?

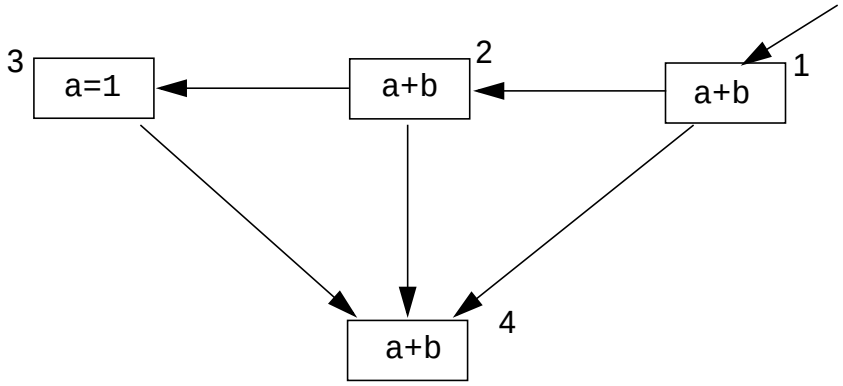
4. Recall that by using constant propagation we can determine that at a particular point in a program a variable v must contain a constant value c. Let us say that at a particular point in a program a variable is **N-limited** if we know that it must contain one of N constant values, c_1, c_2,..., c_n.
   Note that in doing constant propagation, we establish that a variable is 1-limited (limited to one possible value). Boolean variables are naturally 2-limited, since they must always be either true or false.

   (a) (10 points)
   Give examples that illustrate how knowing that a particular variable is N-limited can be used to optimize a program.

   (b) (23 points)
   We can compute N-limited variables by generalizing constant propagation (tracking sets of N values rather than a single value).
   Give a data flow framework for determining N-limited variables for a given N. That is, give the solution lattice, direction, transfer functions and meet operation necessary to compute NLimited(b), where b is a basic block and NLimited(b) is a function that maps a variable name to the set of N values the variable is limited to, or an indication that the variable is not limited to N values.

5. This question involves partial redundancy elimination. The data flow equations that define partial redundancy are listed at the end of this question.

Consider the following control flow graph. What are the PPIn and PPOut values for each block? Where should computations of a+b be added and where should existing computations of a+b be deleted? Explain why your solutions are correct.



$PPOut_b$ = 0 for all exit blocks
$= \underset{k \in succ(b)}{AND} PPIn_k$

$Const_b = AntIn_b$ AND
$[PavIn_b$ OR $(Transp_b$ and $\neg AntLoc_b)]$

$PPIn_b$ = 0 for b_0 (the start block)
$= Const_b$ AND $(AntLoc_b$ or $(Transp_b$ AND $PPOut_b)$
$\underset{p \in pred(b)}{AND} (PPOut_p$ OR $AvOut_p)$

$Insert_b = PPOut_b$ AND $(\neg AvOut_b)$ AND $(\neg PPIn_b$ OR $\neg Transp_b)$
$Remove_b = AntLoc_b$ AND $PPIn_b$

Partial Redundancy Equations

6. (a) (11 points)
Assume we have a C-style "do while" loop of the form

```
do
     body
while (expr);
```

Assume we are doing data flow analysis. Explain how to compute the transfer function of the "do while" loop given the transfer functions of body and expr. Assume that body contains no `goto`s, `break`s, `return`s or `continue`s.

(b) (11 points)
Extend your solution to (a) to the case in which body contains one top-level break controlled by a predicate. That is, the loop is now of the form

```
do
     body1
     if (pred)
              break;
     body2;
while (expr);
```

Explain how to compute the transfer function of this form of loop given the transfer functions of $body_1$, $body_2$, pred and expr. Again, assume that $body_1$ and $body_2$ contain no gotos, breaks, returns or continues.

(c) (11 points)
Extend your solution to (a) to the case in which body contains one top-level continue controlled by a predicate. That is, the loop is now of the form

```
do
     body1
     if (pred)
              continue;
     body2;
while (expr);
```

Explain how to compute the transfer function of this form of loop given the transfer functions of $body_1$, $body_2$, pred and expr. Again, assume that $body_1$ and $body_2$ contain no gotos, breaks, returns or continues.