CS 6340 – Spring 2013 – Assignment 3
Assigned: January 23, 2013
Due: January 30, 2013

Name Sam Britt, Shriram Swaminathan,

Name and Sivaramachandran Ganesan

At the beginning of class on the due date, submit your neatly presented solution with this page stapled to the front (40 points).

NOTE: All work on this problem set is to be done with your partner and without solutions from other past or current students. Any violations will be dealt with according to the Georgia Tech Academic Honor Code and according to the College of Computing process for resolving academic honor code violations. All work must be done using some document creation tool. In addition, graphs must be drawn with a graph-drawing tool—no hand-drawn graphs will be accepted. We'll discuss this requirement more in class.

Given the following program and the statement-based control-flow graph for that program (which you created in Problem Set 1):

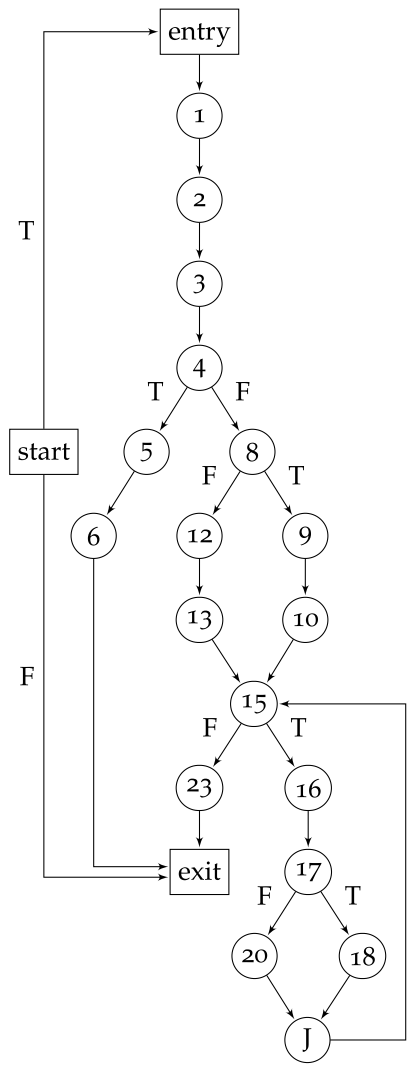
```
      procedure sqrt(real x):real
        real x1,x2,x3,eps,errval;

        begin
   1.     x3 = 1
   2.     errval = 0.0
   3.     eps = .001
   4.     if (x <= 0.0)
   5.        output("illegal operand");
   6.        return errval;
   7.     else
   8.        if (x < 1)
   9.           x1 = x;
   10.          x2 = 1;
   11.       else
   12.          x1 = eps;
   13.          x2 = x;
   14.       endif
   15.       while ( (x2-x1) >= 2.0*eps )
   16.          x3 = (x1+x2)/2.0
   17.          if ( (x3*x3-x)*(x1*x1-x) < 0)
   18.             x2 = x3;
   19.          else
   20.             x1 = x3;
   21.          endif;
   22.       endwhile;
   23.       return x3;
   24.    endif;
   25. end.
```

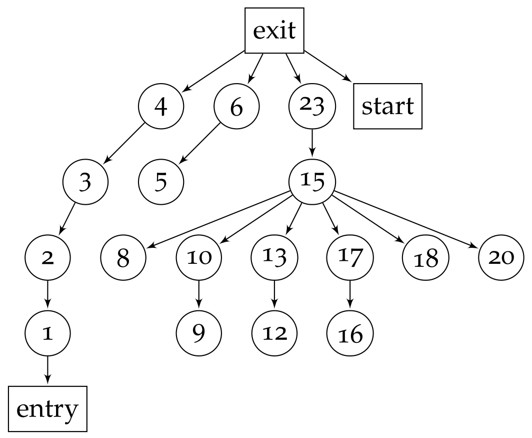
Sam Britt
Shriram Swaminathan
Sivaramachandran Ganesan

CS 6340
Assignment 4
Jan. 30, 2013

1. We start with the control dependence graph, augmented with a "start" node (below, left). From this, we create the postdominator tree (below, right).



Augmented CFG.



Corresponding postdominator tree.

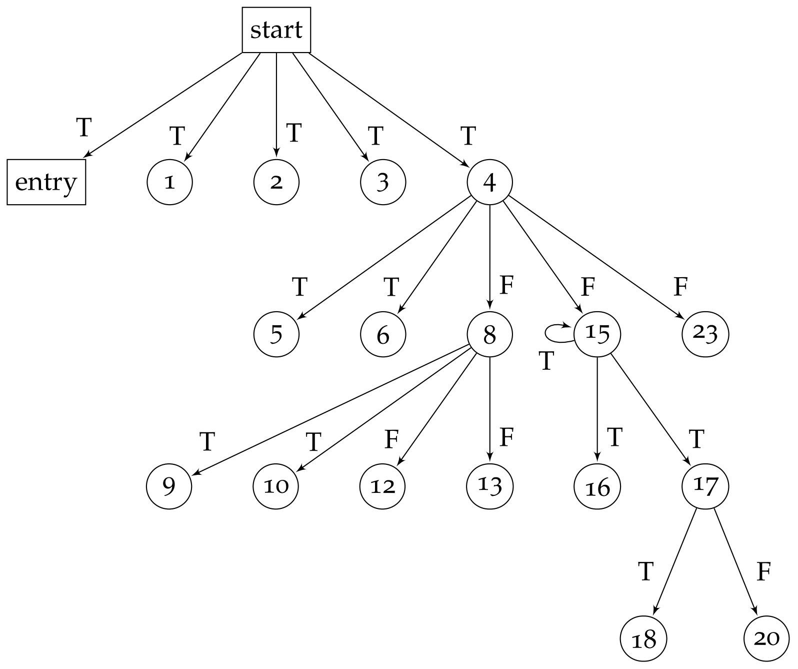
We then find the set $S = \left\{(A,B)\colon (A,B) \in G, \overline{B \operatorname{pdom} A}\right\}$, where G is the above augmented CFG. We find that

$$S = \{(\text{start, entry}), (4, 5), (4, 8), (8, 9), (8, 12), (15, 16), (17, 18), (17, 20)\}.$$

For each edge $(A, B) \in S$, we find L to be the common ancestor of A and B. Finally, the nodes that are control-dependent on A are those the path from L to B on the postdominator tree, including B, and including L only if $L = A$. These results are summarized in the following table. Also noted is "condition," the label on the edge (A, B).

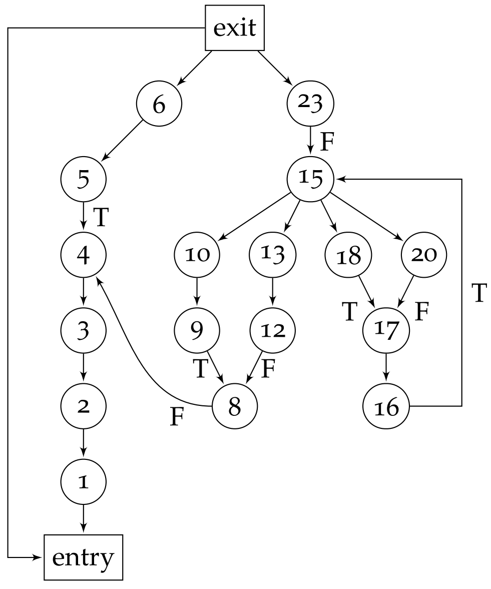
$(A, B) \in S$	Condition (T/F)	L	Nodes dependent on A
(start, entry)	T	exit	{entry, 1, 2, 3, 4}
(4, 5)	T	exit	{5, 6}
(4, 8)	F	exit	{8, 15, 23}
(8, 9)	T	15	{9, 10}
(8, 12)	F	15	{12, 13}
(15, 16)	T	15	{15, 16, 17}
(17, 18)	T	15	{18}
(17, 20)	F	15	{20}

The control dependence graph (below) is constructed directly from the above table.

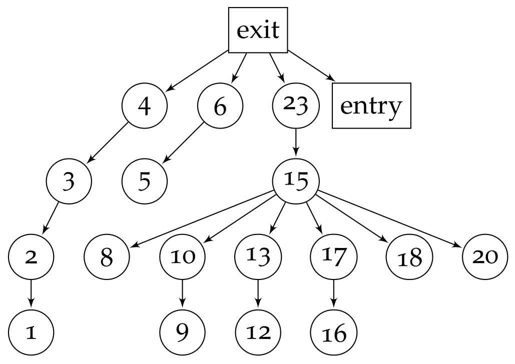


Control dependence graph constructed using the FOW method.

2. We start by augmenting the CFG with an edge from entry to exit, and then reverse the entire graph (below, left). From that we create the dominator tree (below, right).



Reverse CFG.

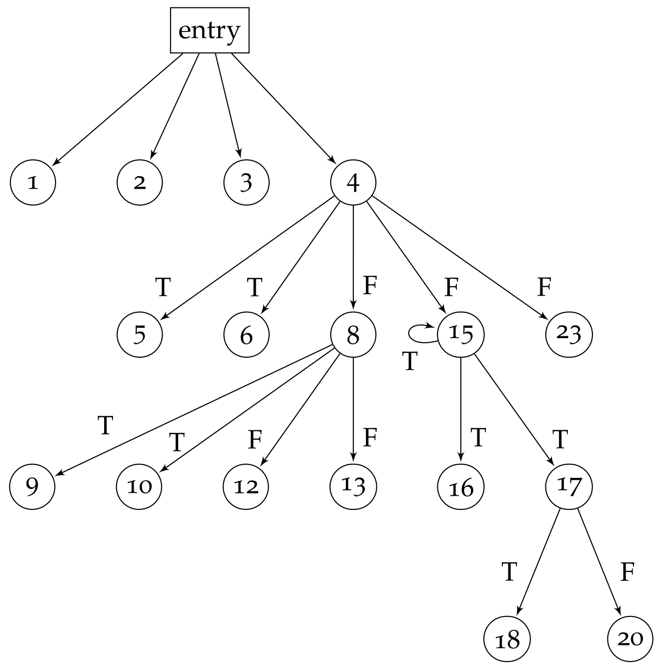


Corresponding dominator tree.

For each node n, we traverse down the reverse CFG (RCFG) until we find a node that n does not strictly dominate; that is, a node that is a child of n in the RCFG but not a child of n in the dominator tree. This set of nodes is the dominance frontier of n. These are tabulated below. Also included, in subscripts where appropriate, is the *condition* of the control dependence. If node v is in the dominance frontier of n, then the condition of v is the label on the edge (u, v) where $n \operatorname{dom} u$.

Block	Dominance frontier
entry	$\emptyset$
1	{entry}
2	{entry}
3	{entry}
4	{entry}
5	$\{4_T\}$
6	$\{4_T\}$
8	$\{4_F\}$
9	$\{8_T\}$
10	$\{8_T\}$
12	$\{8_F\}$
13	$\{8_F\}$
15	$\{4_F, 15_T\}$
16	$\{15_T\}$
17	$\{15_T\}$
18	$\{17_T\}$
20	$\{17_F\}$
23	$\{4_F\}$

By inverting the dominance frontier sets, we arrive at the control dependence graph below. Edges are labeled with the appropriate conditions.



Control dependence graph constructed using the dominance frontier method.

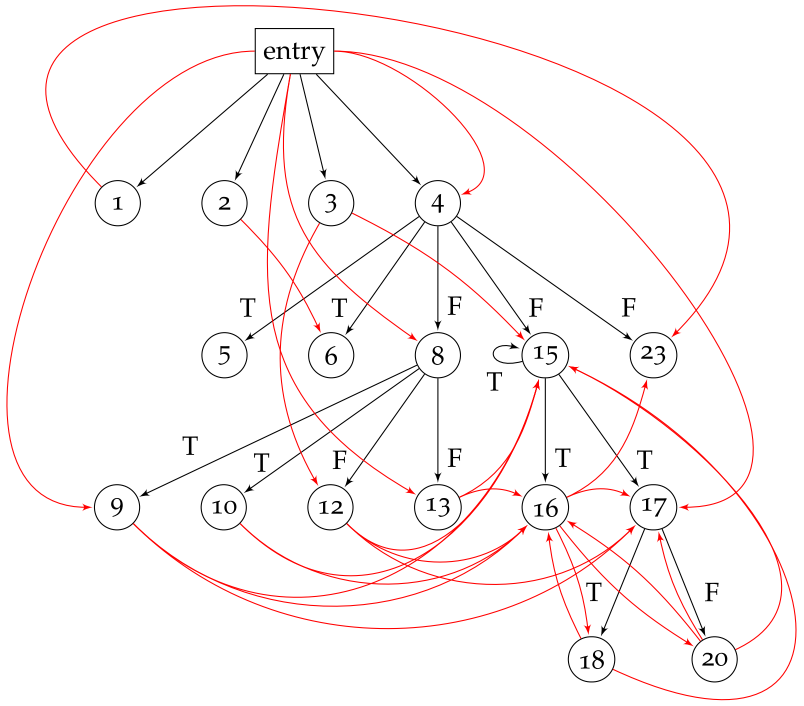
To add the data dependence subgraph, we need the definition-use pairs. These were computed in Assignment 3; the results are repeated in the following table. In the following, if a source code line contains only one use, the use is denoted by the line number. Otherwise, the notation $\ell.i$ means the ith use in line ℓ. Definitions in the entry block (that is, program parameters) are denoted by E.

Use	Variable	Reaching Definitions
4	`x`	$\{E\}$
6	`errval`	{2}
8	`x`	$\{E\}$
9	`x`	$\{E\}$
12	`eps`	{3}
13	`x`	$\{E\}$
15.1	`x2`	{10, 13, 18}
15.2	`x1`	{9, 12, 20}
15.3	`eps`	{3}
16.1	`x1`	{9, 12, 20}
16.2	`x2`	{10, 13, 18}
17.1	`x3`	{16}
17.2	`x3`	{16}
17.3	`x`	$\{E\}$
17.4	`x1`	{9, 12, 20}
17.5	`x1`	{9, 12, 20}
17.6	`x`	$\{E\}$
18	`x3`	{16}
20	`x3`	{16}
23	`x3`	{1, 16}

By inverting the reaching definitions sets, we arrive at the definition-use data dependencies, tabulated below.

Definition	Variable	Reachable Uses
E	`x`	{4, 8, 9, 13, 17.3, 17.6}
1	`x3`	{23}
2	`errval`	{6}
3	`eps`	{12, 15.3}
9	`x1`	{15.2, 16.1, 17.4, 17.5}
10	`x2`	{15.1, 16.2}
12	`x1`	{15.2, 16.1, 17.4, 17.5}
13	`x2`	{15.1, 16.2}
16	`x3`	{17.1, 17.2, 18, 20, 23}
18	`x3`	{15.1, 16.2}
20	`x3`	{15.2, 16.1, 17.4, 17.5}

Finally, we augment the CDG, drawing an edge from node A to B if A contains a definition of some variable v, B contains a use of v, and B is reachable by A. Edges representing data dependencies are drawn in red in the PDG below.



Program dependence graph. Control dependencies are in black, and data dependencies are in red.