

# Assignment 3

## Dataflow analysis and Register allocation

Due: February 8, 11:00 am

This is a written assignment, every student must hand in his or her homework. Bring your homework to class on February 8th. SCPD students may submit their homework by e-mail via [scpd-distribution@lists.stanford.edu](mailto:scpd-distribution@lists.stanford.edu) or give your homework to the courier.

1.a. We track the signs we will design a forward data flow analysis. The lattice elements are an assignment to each variable  $x \in V$  of an element of the power set  $2^S$ , where  $S$  is the set of signs  $\{+, -, 0\}$ . The values in this set represent ranges of values the variable might have, so that the sign of a variable can be determined from the value it has in the lattice element, e.g.  $\{0, +\}$  means the variable is non-negative.

Direction	forward
Values	$F : V \rightarrow 2^S$
Meet	$(F_1 \wedge F_2)(x) = F_1(x) \cup F_2(x)$
Top	$F: x \mapsto \{\}$
Bottom	$F: x \mapsto \{+, -, 0\}$
Entry	$\perp$
Transfer function	See below
Monotonicity	Yes
Distributivity	No

For addition, subtraction and multiplication we will use functions  $f_+$ ,  $f_-$ , and  $f_*$  with the signature  $S \times S \rightarrow 2^S$ . These functions are defined in the tables below;  $f_+(s, s')$  gives the possible signs of the result of adding two values with signs  $s$  and  $s'$ , and similarly for  $f_-$  and  $f_*$ .

Now, for  $x = y + z$ , if the possible signs of  $y$  and  $z$  are  $F(y)$  and  $F(z)$  for the previous dataflow value  $F$  (if  $y$  or  $z$  are constants just use the sign for that constant), then the new signs for  $x$  are:

$$F'(x) = \bigcup_{s \in F(y), s' \in F(z)} f_+(s, s')$$

And similarly for the subtraction and multiplication operations. The functions  $f_+$ ,  $f_-$  and  $f_*$  are defined as follows.  $f_+(s, s')$  can be found by looking up the row for  $s$  and column for  $s'$ .

$f_+$	+	-	0
+	$\{+\}$	$\{+,-,0\}$	$\{+\}$
-	$\{+,-,0\}$	$\{-\}$	$\{-\}$
0	$\{+\}$	$\{-\}$	$\{0\}$

$f_-$	+	-	0
+	{+,-,0}	{+}	{+}
-	{-}	{+,-,*}	{-}
0	{-}	{+}	{0}

$f_*$	+	-	0
+	{+}	{-}	{0}
-	{-}	{+}	{0}
0	{0}	{0}	{0}

This framework is monotone but not distributive. Monotonicity follows from the design of the transfer function; if the input lattice values are larger, the output value will be larger. A counterexample for distributivity is as follows:

```

if (...) {
  x = 1;
  y = 1;
}
else {
  x = -1;
  y = -1;
}
z = x * y;

```

The MOP solution will determine  $z$  is positive ( $\{+\}$ ) at the end, while the MFP solution will determine  $z$  is non-zero ( $\{+, -\}$ ).

b. To handle **assume** statements, we just need to add new transfer functions for these, which have the effect of removing entries from  $F(x)$  for variables  $x$ . Define new functions  $f_>$ ,  $f_\geq$ ,  $f_<$  and  $f_\leq$  with signatures  $S \rightarrow 2^S$ .  $f_>(s)$  gives the possible signs of a value which is greater than some value with sign  $s$ , and similarly for  $f_\geq$ ,  $f_<$  and  $f_\leq$ .

Now, for `assume(x < y)`, we will restrict the possible signs of  $x$  and  $y$  as follows:

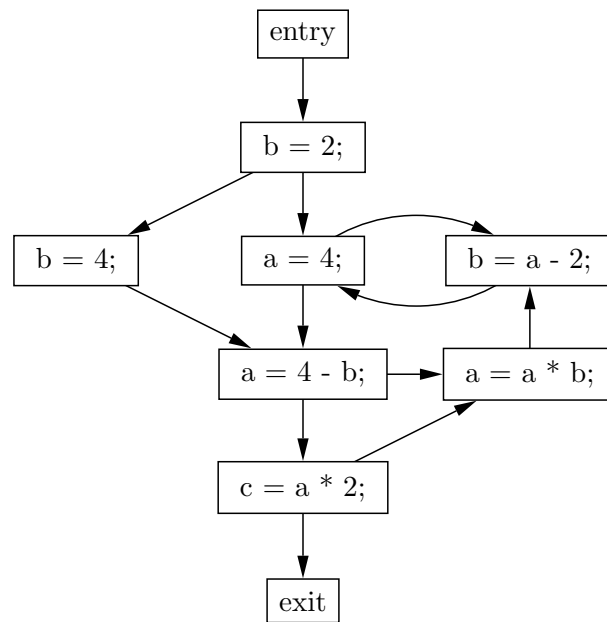
$$F'(x) = F(x) \cap \bigcup_{s \in F(y)} f_<(s)$$

$$F'(y) = F(y) \cap \bigcup_{s \in F(x)} f_>(s)$$

And similarly for other assumes. The functions  $f_>$ ,  $f_\geq$ ,  $f_<$  and  $f_\leq$  are defined as follows.  $f_>(s)$  can be found by looking up the row for  $s$  and column for  $f_>$ .

	$f_>$	$f_\geq$	$f_<$	$f_\leq$
+	{+}	{+}	{+,-,0}	{+,-,0}
-	{+,-,0}	{+,-,0}	{-}	{-}
0	{+}	{+,0}	{-}	{-,0}

2. The following graph shows the result of constant propagation. The only changes are to the 'a = b \* 2', 'c = a \* b', and 'c = a \* 2' statements.



3.

a. Any of the following possible reverse post-orderings:

Entry A B E F D C G H X

Entry A B E F G H X D C

Entry A B E G H F D C X

Entry A B E G H X F D C

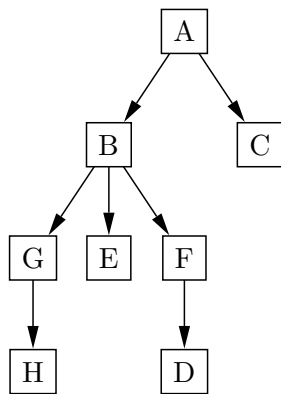
Entry A C B E F D G H X

Entry A C B E F G H X D

Entry A C B E G H F D X

Entry A C B E G H X F D

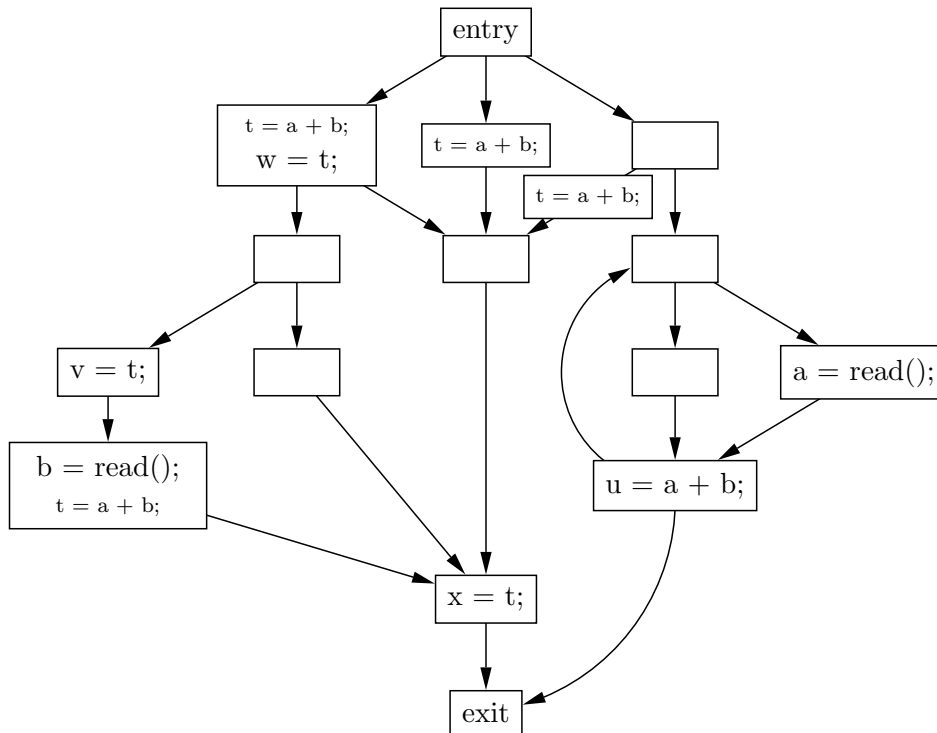
b. Dominator tree:

c. Back Edges:  $G \rightarrow B$ ,  $F \rightarrow A$ Natural Loop for  $G \rightarrow B$ :  $\{B, E, F, G, H\}$ Natural Loop for  $F \rightarrow A$ :  $\{A, B, C, D, E, F, G, H\}$ 

d. The graph is *not reducible*.  $\{H \rightarrow F, C \rightarrow B\}$  are retreating edges but not back edges. Equivalently, removing back edges  $\{G \rightarrow B, F \rightarrow A\}$  does not eliminate all cycles in the graph.

e. Remove either  $\{A \rightarrow C, H \rightarrow F\}$  or  $\{A \rightarrow B, H \rightarrow F\}$

4.



5. Yes, it is possible to find an assignment. One possible assignment is: R1: { A, B, I, G}, R2:{ C, F, D}, R3:{ E, H}. The algorithm we mentioned in class will stuck after removing C, H, G.