

CSE 401 - Section 9 - Adventures in Dataflow Analysis - Solutions

1. **Reaching Definitions** Consider the following small program that we used as a dataflow example for live variable analysis in lecture. This time all of the statements are labeled individually and we want to compute reaching definitions.

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

The reaching definitions dataflow problem is to determine for each variable definition which other blocks in the control flow graph could potentially see the value of the variable that was assigned in that definition. **To simplify things, we will treat each individual statement above as a separate block, and use the statement labels as the names of both the blocks and the definitions in them.** So, for example, reaching definition analysis would allow us to determine that definition L0, which assigns to *a*, can reach block L1.

A definition *d* in block *p* reaches block *q* if there is at least one path from *p* to *q* along which definition *d* is not redefined.

- a) Come up with a formulation of this analysis as a dataflow problem by describing the following basic dataflow sets in terms of this task (you may use informal descriptions if your meaning is clear).

GEN(*b*): the definitions assigned and not killed in block *b*

KILL(*b*): the definitions of variables overwritten in block *b*

IN(*b*): the definitions that are reaching upon entering block *b*

OUT(*b*): the definitions that are reaching upon exiting block *b*

- b) Define the relationship between the sets by giving equations for IN(*b*) and OUT(*b*) in terms of other sets and other basic blocks as needed.

$$\text{IN}(b) = \bigcup_{p \in \text{pred}(b)} \text{OUT}(p)$$

$$\text{OUT}(b) = \text{GEN}(b) \cup (\text{IN}(b) - \text{KILL}(b))$$

- c) Compute the reaching definitions for the blocks in the given program, treating each statement as a separate block. In the following table, compute the GEN and KILL sets for each block, and then use those answers to compute successive iterations of the IN and OUT sets until there are no more changes to be made.

Note that this is a forward dataflow analysis problem, so the answer will converge faster if you compute from beginning to end (i.e. starting with L_0).

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L_0	L_3		L_0		L_0
L1	L_1		L_0	L_0, L_1	L_0, L_1, L_2, L_3	L_0, L_1, L_2, L_3
L2	L_2		L_0, L_1	L_0, L_1, L_2	L_0, L_1, L_2, L_3	L_0, L_1, L_2, L_3
L3	L_3	L_0	L_0, L_1, L_2	L_1, L_2, L_3	L_0, L_1, L_2, L_3	L_1, L_2, L_3
L4			L_1, L_2, L_3	L_1, L_2, L_3	L_1, L_2, L_3	L_1, L_2, L_3
L5			L_1, L_2, L_3	L_1, L_2, L_3	L_1, L_2, L_3	L_1, L_2, L_3

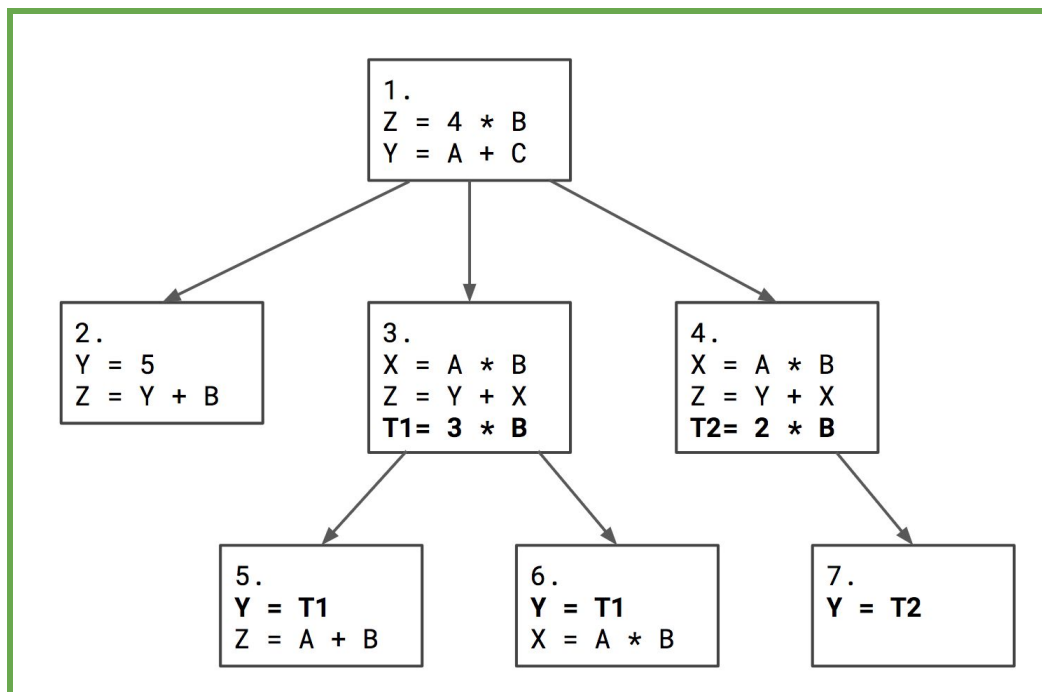
- d) Now that we have completed our dataflow analysis, we want to apply optimizations to the code. After noticing that the definition L_0 of the variable a is a constant value, we wonder if it is possible to use constant propagation to replace uses of the variable a with the constant 0.

Is it possible to replace the use of a in block L1 with the constant 0? Justify your answer using evidence from the sets that you computed during dataflow analysis.

No, it is not possible. To determine this, we would look at the IN set for block L1 -- the fact that the IN set contains two definitions of 'a' (L_0 and L_3) means we cannot perform this constant propagation. In other words, more than one definition of 'a' is a reaching definition to block L1, and therefore performing constant propagation would only preserve one possible value of 'a' and the generated code would not be equivalent.

2. **Very Busy Expressions** In the following Control Flow Graph, rearrange the expressions among the blocks to eliminate Very Busy expressions (perform code hoisting). In other words, come up with the earliest location that each expression can be computed without affecting the outcome of the code. For this problem, you do not need to formally run through any dataflow analyses -- it is enough to reason through the code on your own to determine what Very Busy expressions exist and how they can be hoisted.

a) Draw the new control flow graph.



b) What is the benefit of this optimization? Does it make the code more efficient?

Solution: In general, this optimization does not improve the efficiency of the code at all. Its benefit is that it reduces the size of the generated code.