CS 243 - Section 1

Winter 2021

January 15, 2021

1 Anticipation

We call an expression A *op* B "anticipated" at a program point p if along every path from p there is an expression A *op* B that shows up before either A or B is redefined.

This notion is an integral part of partial redundancy elimination (PRE), since it determines the earliest point to which expressions can be moved without resulting in redundant or dead computations. It is in fact the first of several dataflow analyses that make up PRE.

Suppose our language only contains two types of expressions: `x = y`, `x = y + z`.

Design a dataflow algorithm to determine the set of expressions (in the form of `a + b`) that are anticipated at each program point. You may assume that each basic block is a single statement.

Direction of your analysis (forward/backward)	Backwards.
Lattice elements and meaning	Set of expressions `a + b`
Meet operator or lattice diagram	$\cap$
Is there a top element? If yes, what is it?	U.
Is there a bottom element? If yes, what is it?	$\emptyset$.
Transfer function of a basic block	Kill[B] = { a + b | a or b is redefined in B } Gen[B] = { a + b | a + b is assigned to a variable in B } `x = y` | In[b] = Out[B] - Kill[B] `x = y + z` | In[b] = (Out[B] - Kill[B]) $\cup$ Gen[B]
Boundary condition initialization	$\emptyset$.
Interior points initialization	all expressions.