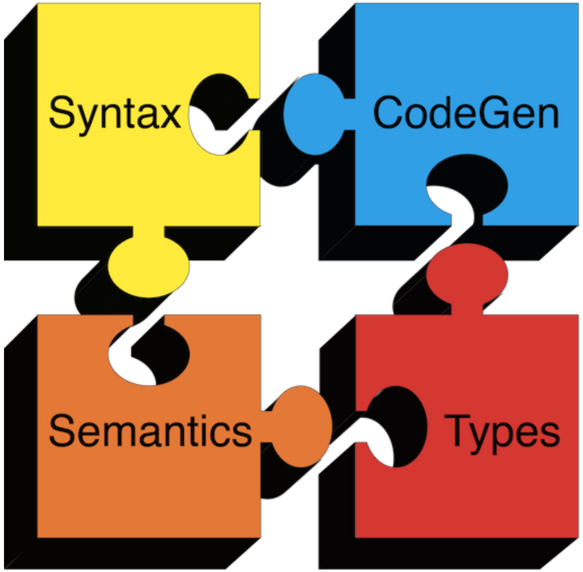


Compilers

Constant Propagation

Alex Aiken

Constant Propagation

To replace a use of x by a constant k we must know:

On every path to the use of x, the last assignment to x is
x := k **

Constant Propagation

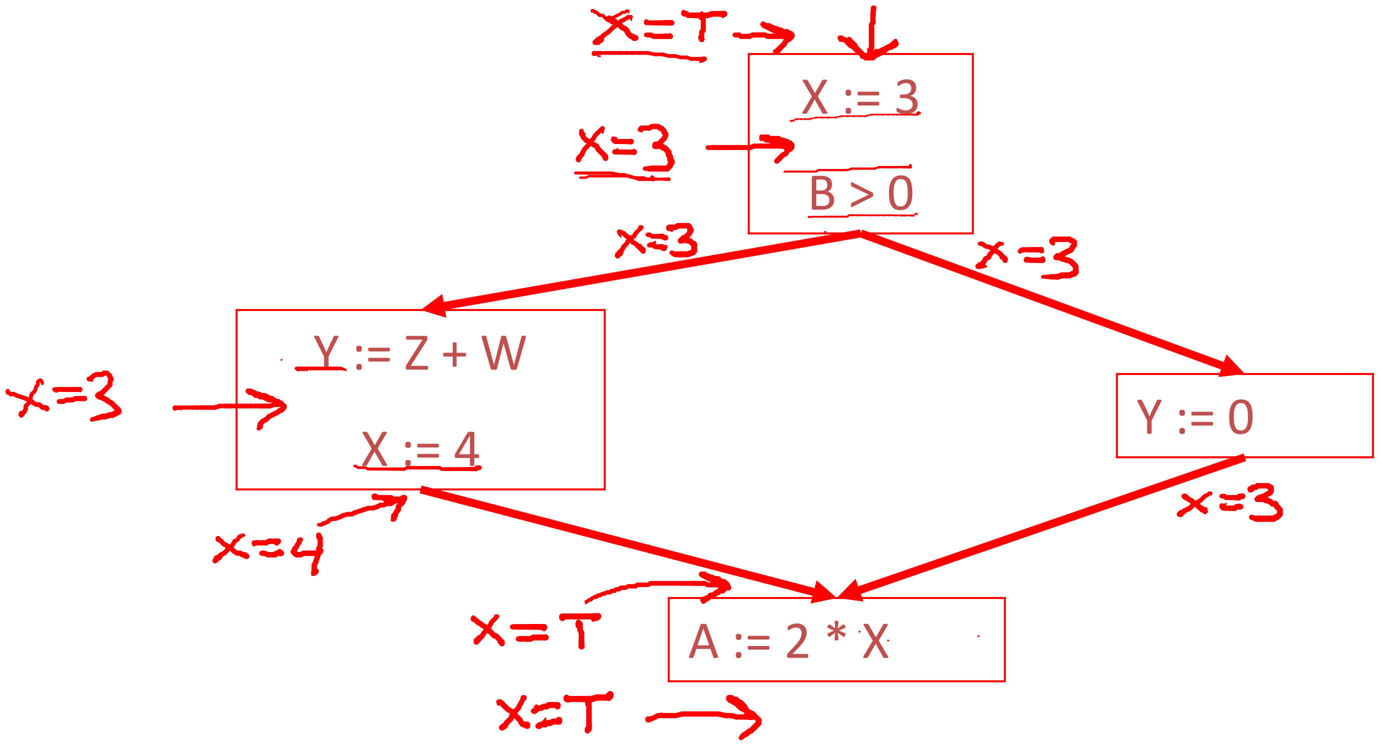
- Global constant propagation can be performed at any point where ** holds

- Consider the case of computing ** for a single variable X at all program points

- To make the problem precise, we associate one of the following values with X at every program point

value	*interpretation*
$\perp$ "bottom"	This statement never executes
c	X = constant c
T "top"	X is not a constant

Constant Propagation



Constant Propagation

- Given global constant information, it is easy to perform the optimization
  - Simply inspect the x = ? associated with a statement using x
  - If x is constant at that point replace that use of x by the constant

- But how do we compute the properties x = ?

The analysis of a complicated program can be expressed as a combination of simple rules relating the change in information between adjacent statements

Constant Propagation

- The idea is to "push" or "transfer" information from one statement to the next

- For each statement s, we compute information about the value of x immediately before and after s

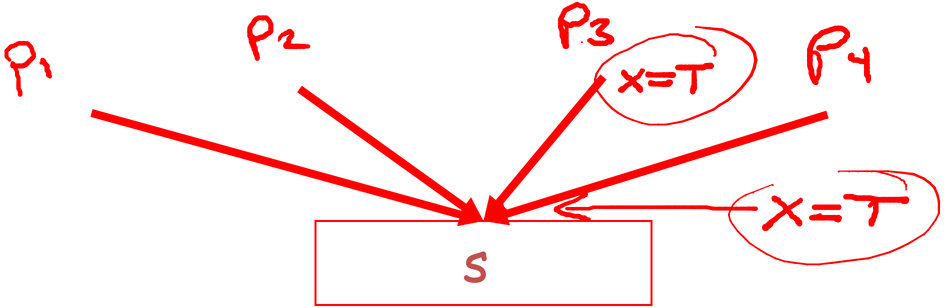
$$C(x,s,in) = \text{value of } x \text{ before } s$$

$$C(x,s,out) = \text{value of } x \text{ after } s$$

- Define a *transfer* function that transfers information one statement to another

- In the following rules, let statement s have immediate predecessor statements $p_1, \ldots, p_n$

Rule 1

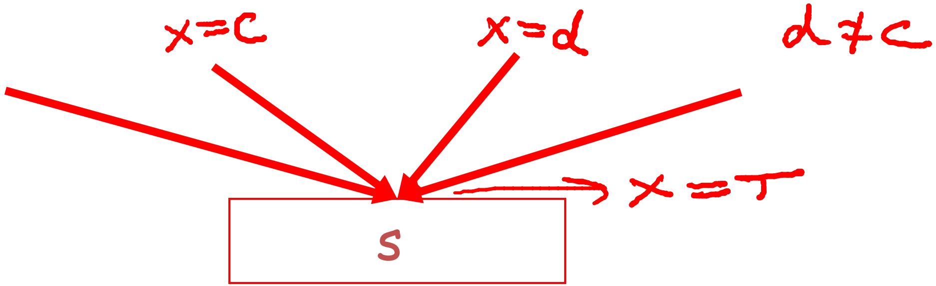


if $C(p_i, x, out) = T$

for any i, then $C(s, x, in) = T$

Constant Propagation

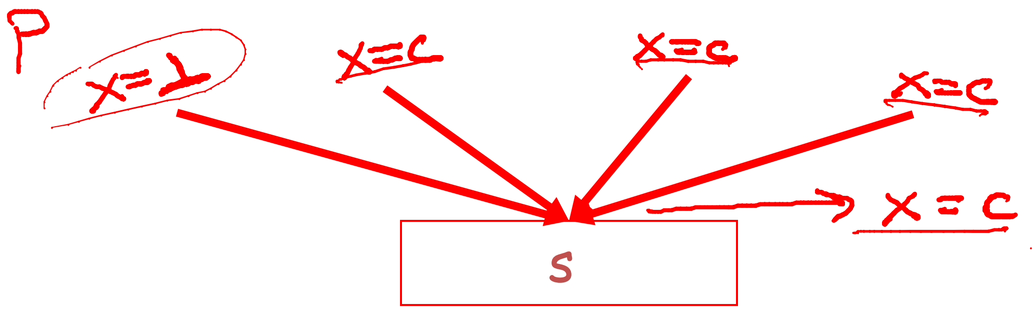
Rule 2



if $C(p_i, x, out) = c$ & $C(p_j, x, out) = d$ & $d <> c$

then $C(s, x, in) = \top$

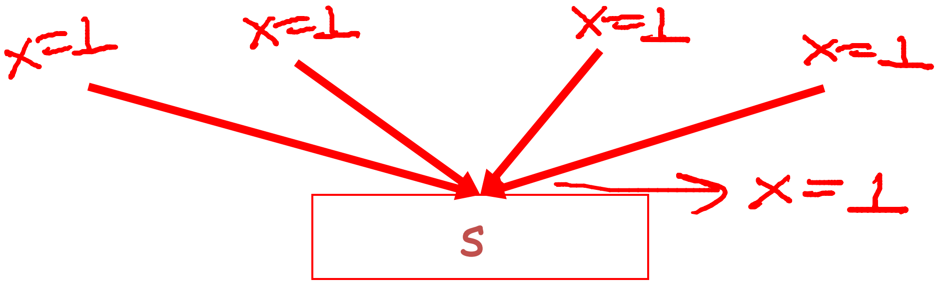
Rule 3



if $C(p_i, x, out) = c$ or $\perp$ for all i,

then $C(s, x, in) = c$

Rule 4



if $C(p_i, x, out) = \perp$ for all i,

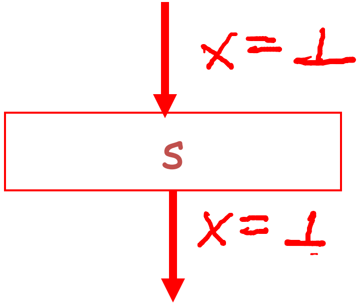
then $C(s, x, in) = \perp$

Constant Propagation

- Rules 1-4 relate the *out* of one statement to the *in* of the next statement

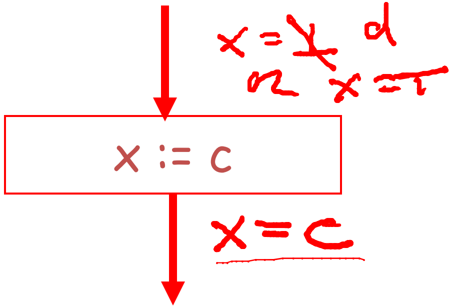
- Now we need rules relating the *in* of a statement to the *out* of the same statement

Rule 5



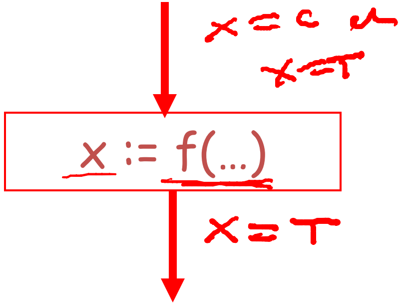
$$C(s, x, out) = \perp \ \text{ if } C(s, x, in) = \perp$$

Rule 6



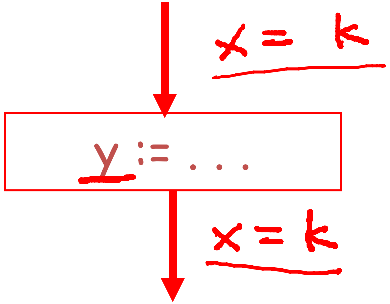
$$C(x := c, x, out) = c \text{ if } c \text{ is a constant}$$

Rule 7



$$C(x := f(\ldots), x, out) = \top$$

Rule 8



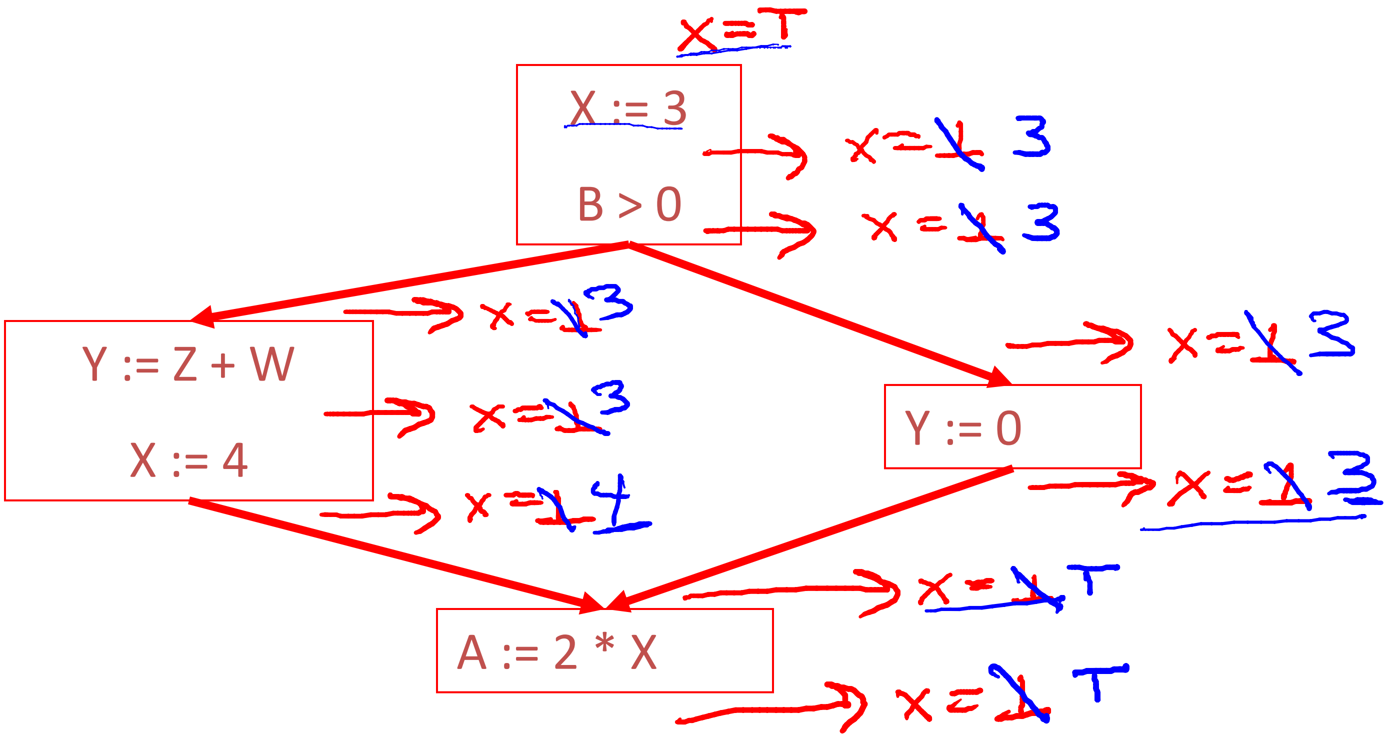
C(y := …, x, out) = C(y := …, x, in) if x <> y

Constant Propagation

1. For every entry s to the program, set $C(s, x, in) = \top$
2. Set $C(s, x, in) = C(s, x, out) = \bot$ everywhere else
3. Repeat until all points satisfy 1-8:

   Pick s not satisfying 1-8 and update using the appropriate rule

Constant Propagation



Constant Propagation

After running the constant propagation algorithm to completion, choose the correct dataflow information for X, Y, and Z at the program point labeled at right.

	X	Y	Z
○	4	T	T
○	4	T	5
○	4	1	5
○	T	T	T