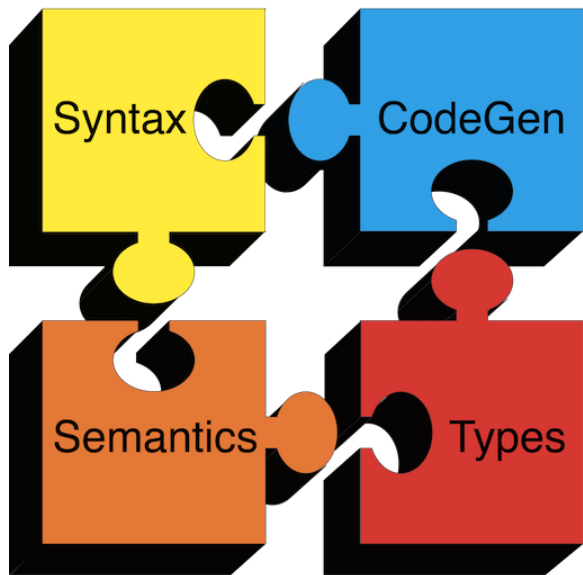




Compilers

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

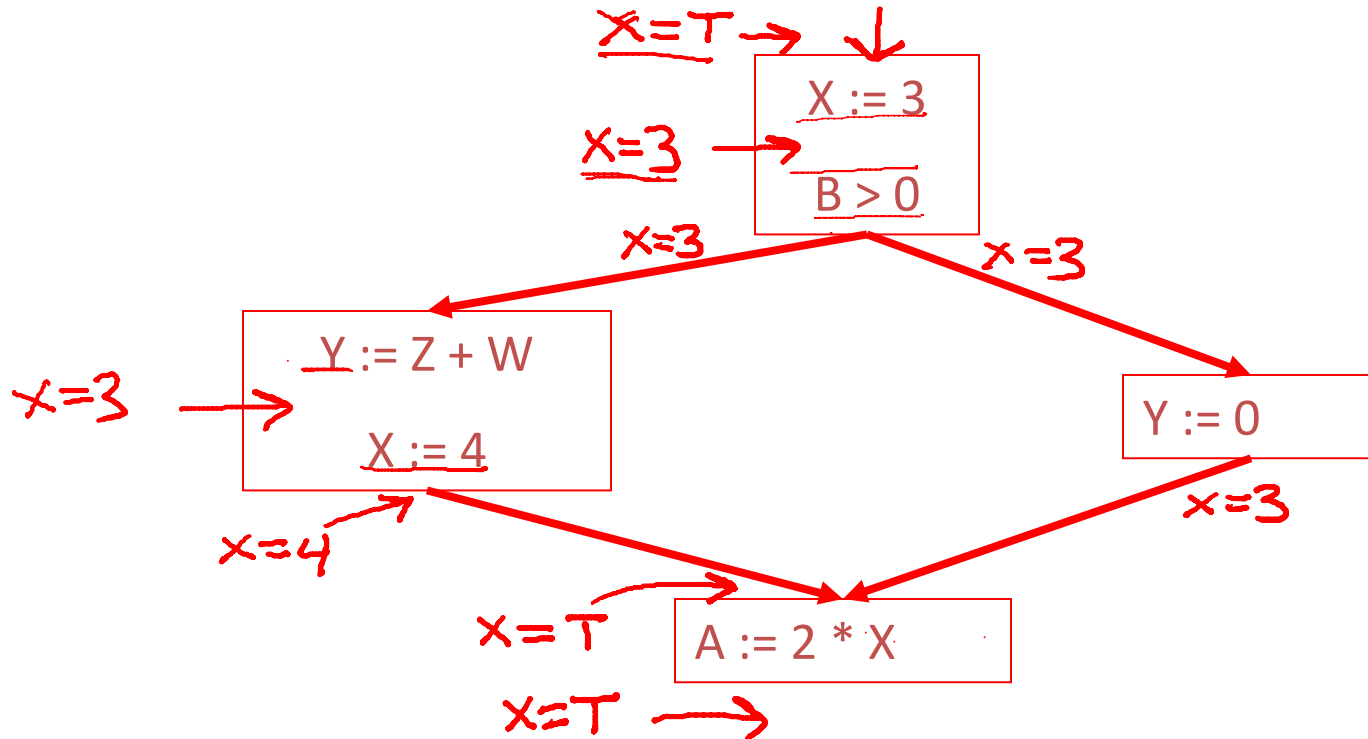
$x := k$ **

- 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

	<i>value</i>	<i>interpretation</i>
→	$\perp$ "bottom"	<u>This statement never executes</u>
→	c	$X = \text{constant } c$
→	T "top"	X is not a constant

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

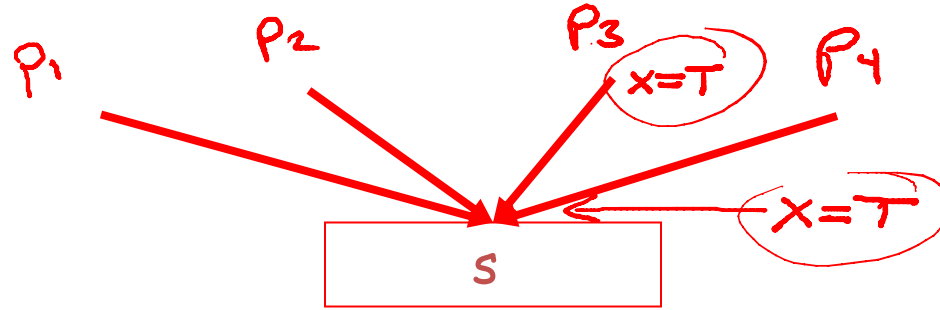
- The idea is to “push” or “transfer” information from one statement to the next
- For each statement s , we compute information about the value of x immediately before and after s

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

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

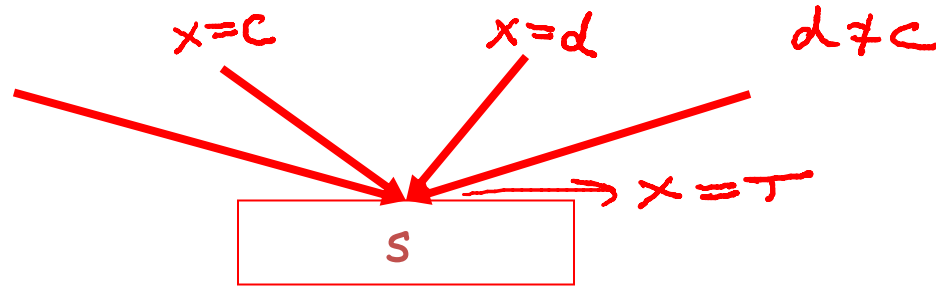
- Define a transfer function that transfers information one statement to another
- In the following rules, let statement s have immediate predecessor statements $p_1, \dots, p_n$

Rule 1



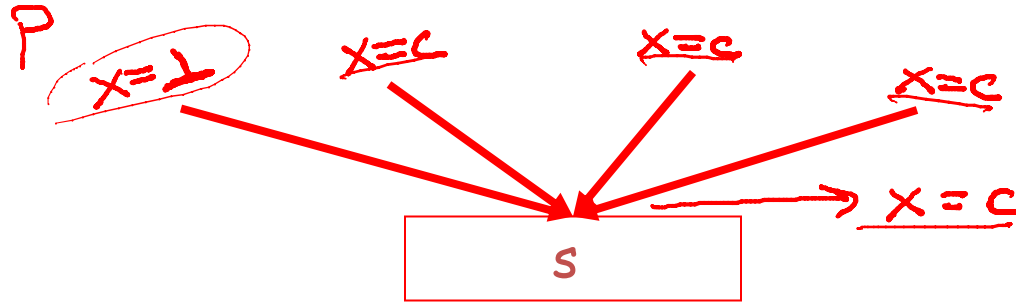
if $C(\underline{p_i}, \underline{x}, \underline{\text{out}}) = \underline{T}$
for any i , then $C(\underline{s}, \underline{x}, \underline{\text{in}}) = \underline{T}$

Rule 2



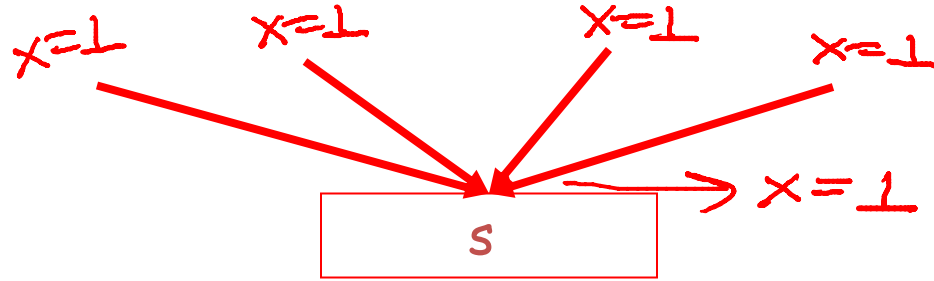
if $C(p_i, x, \text{out}) = c$ & $C(p_j, x, \text{out}) = d$ & $d \neq c$
then $C(s, x, \text{in}) = \top$

Rule 3



if $C(p_i, x, \text{out}) = c$ or $\perp$ for all i ,
then $C(s, x, \text{in}) = c$

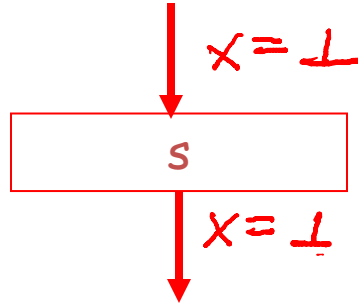
Rule 4



if $C(p_i, x, \text{out}) = \perp$ for all i ,
then $C(s, x, \text{in}) = \perp$

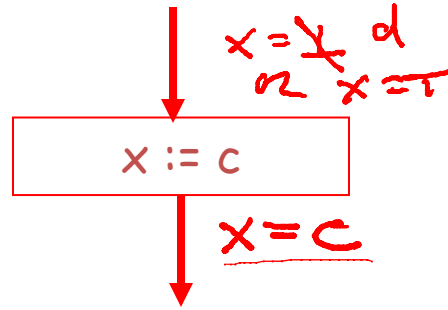
- Rules 1-4 relate the out of one statement to the in of the next statement
- Now we need rules relating the in of a statement to the out of the same statement

Rule 5



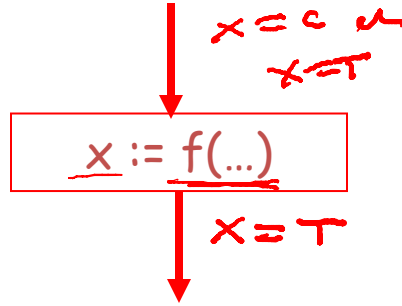
$$C(s, x, \text{out}) = \perp \text{ if } C(s, x, \text{in}) = \perp$$

Rule 6



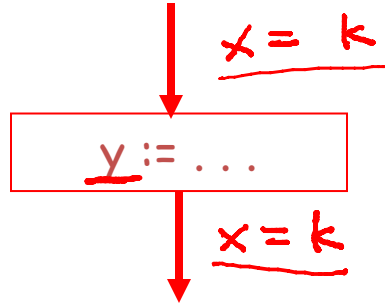
$C(x := c, x, \text{out}) = c$ if c is a constant

Rule 7



$$C(x := f(\dots), x, \text{out}) = T$$

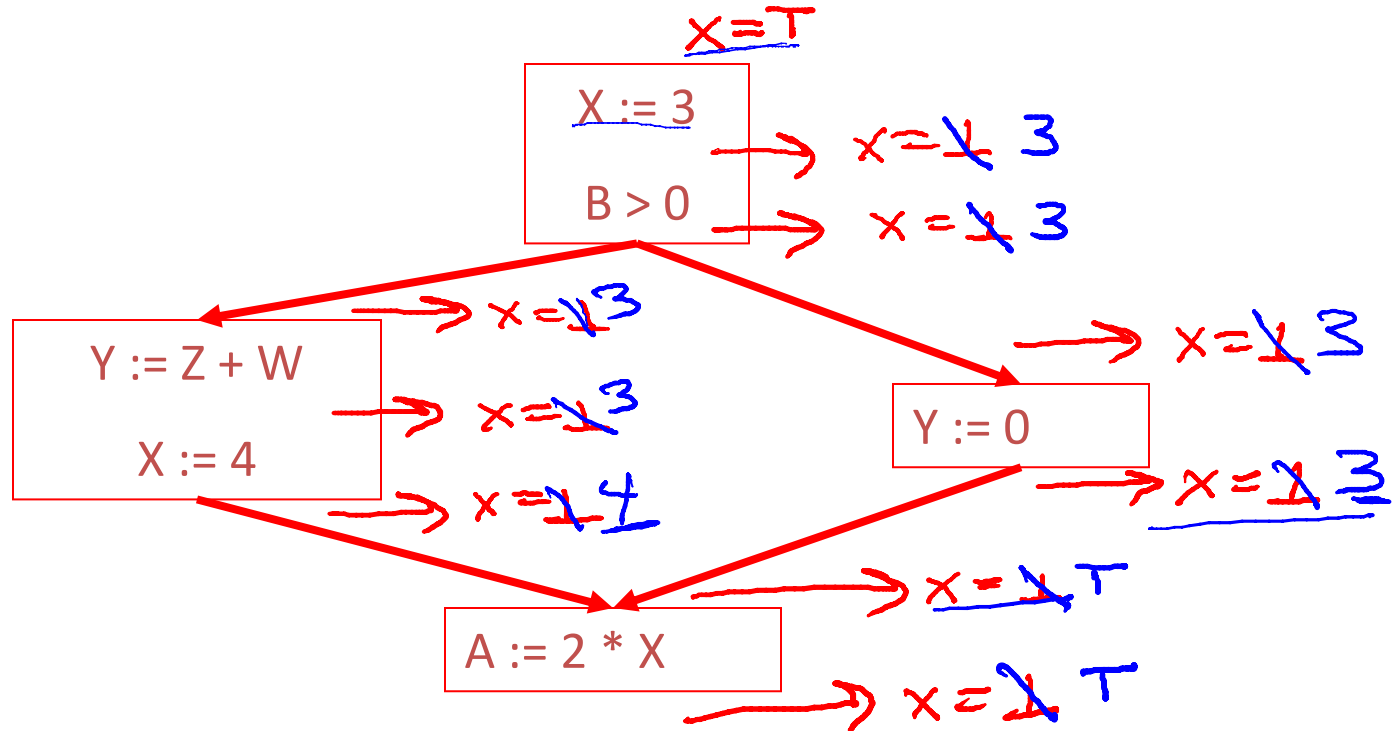
Rule 8



$$C(y := \dots, x, \text{out}) = C(y := \dots, x, \text{in}) \text{ if } x \neq y$$

1. For every entry s to the program, set $C(s, x, \text{in}) = \text{T}$
2. Set $C(s, x, \text{in}) = C(s, x, \text{out}) = \perp$ 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

