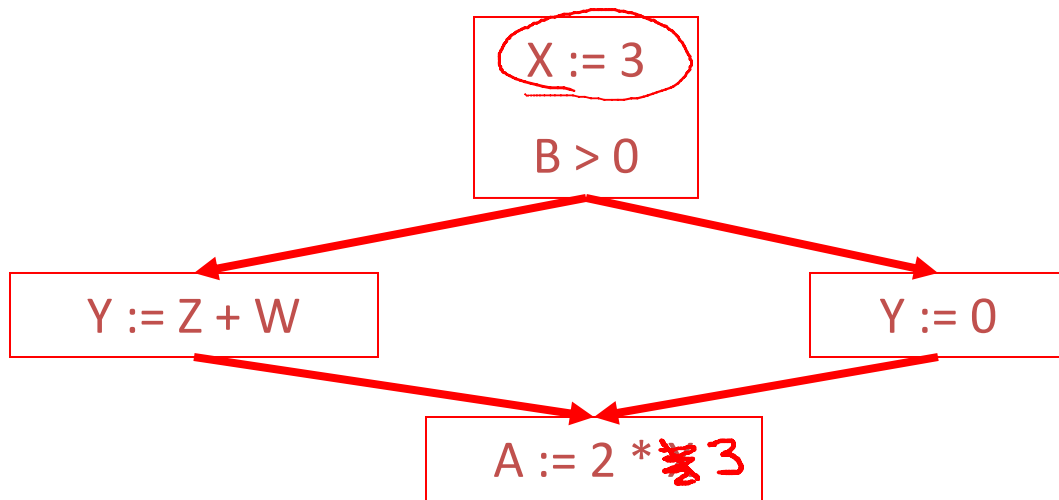




Compilers

Liveness Analysis

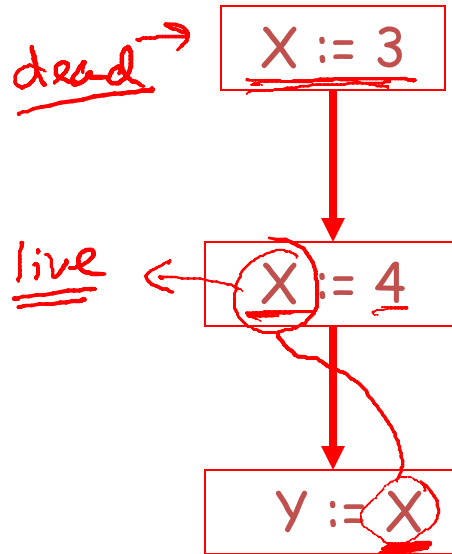
Once constants have been globally propagated, we would like to eliminate dead code



After constant propagation, `X := 3` is dead
(assuming `X` not used elsewhere)

Liveness Analysis

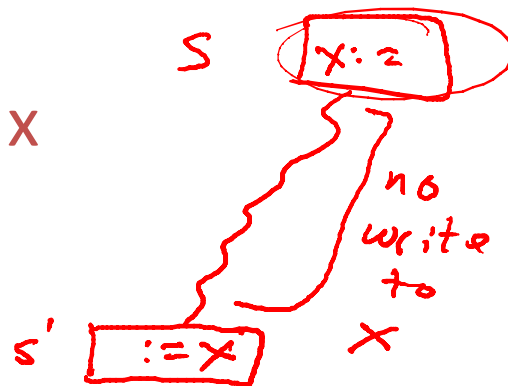
- The first value of x is *dead* (never used)
- The second value of x is *live* (may be used)
- Liveness is an important concept



live =
may be
used in
the future

A variable x is live at statement s if

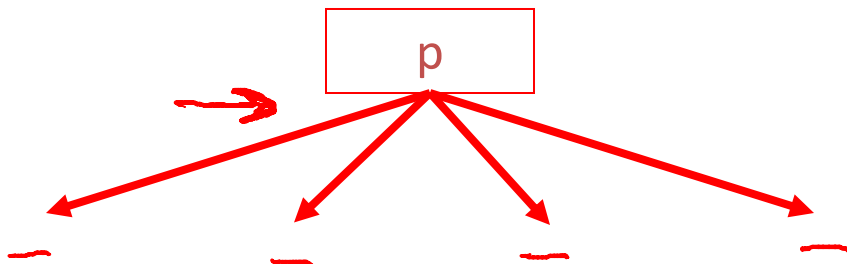
- There exists a statement s' that uses x
- There is a path from s to s'
- That path has no intervening assignment to x



- A statement $x := \dots$ is dead code if x is dead after the assignment
- Dead statements can be deleted from the program
- But we need liveness information first . . .

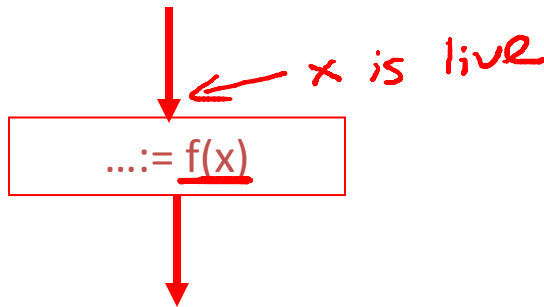
- We can express liveness in terms of information transferred between adjacent statements, just as in copy propagation
- Liveness is simpler than constant propagation, since it is a boolean property (true or false)

Rule 1



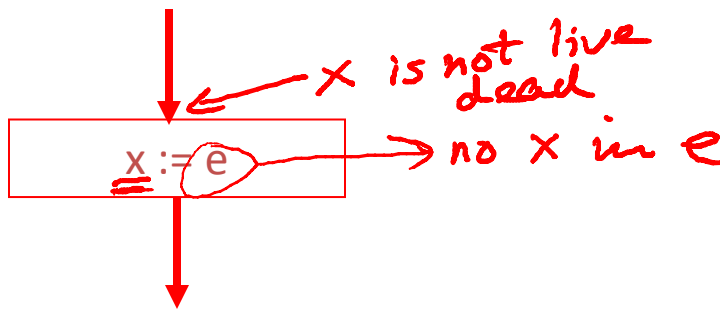
$$L(p, x, \text{out}) = \underbrace{\vee \{ L(s, x, \text{in}) \mid s \text{ a successor of } p \}}$$

Rule 2



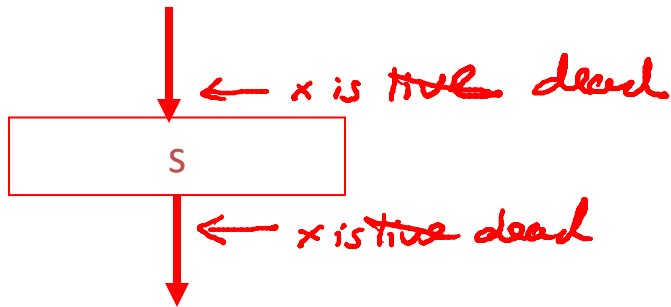
$L(s, x, \underline{\text{in}}) = \underline{\text{true}}$ if s refers to x on the rhs

Rule 3



$L(x := e, x, in) = \text{false}$ if e does not refer to x

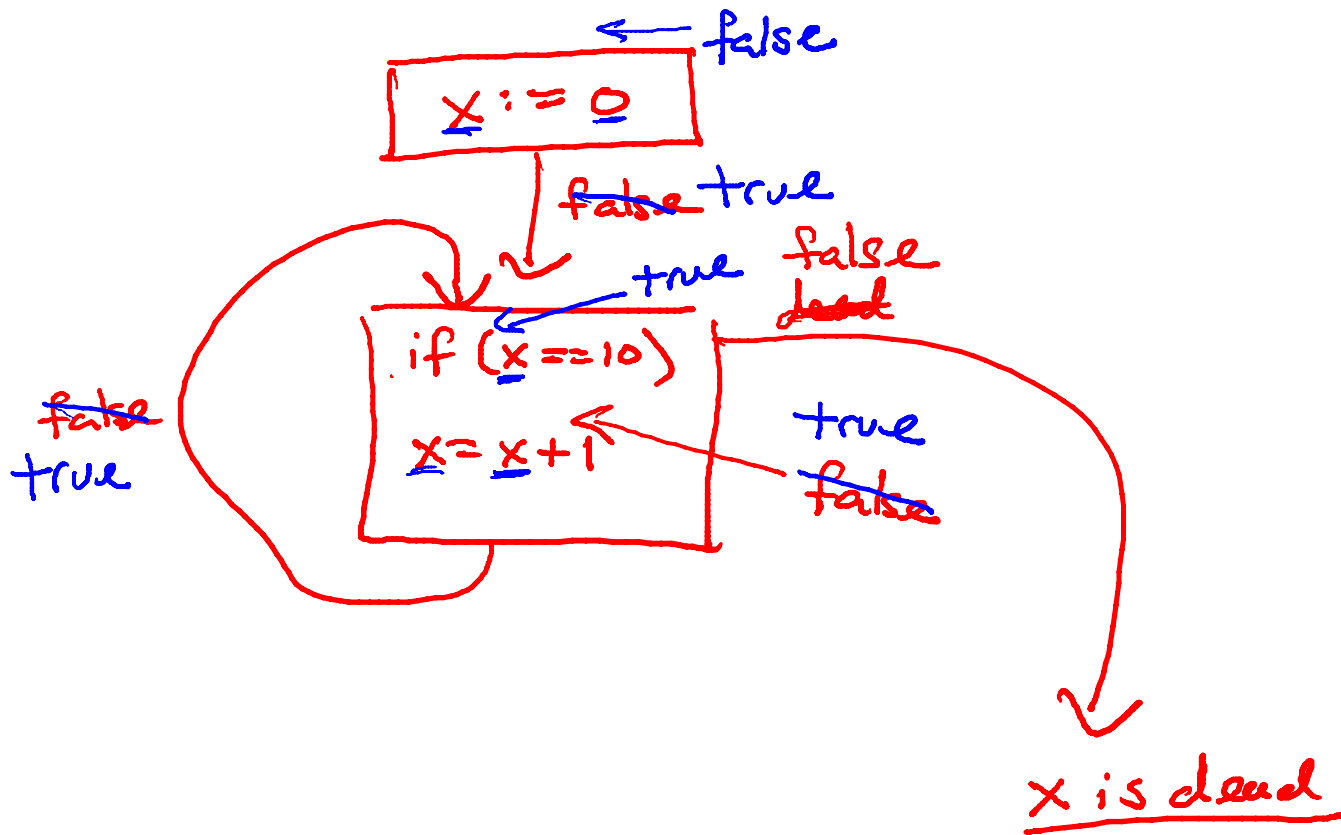
Rule 4



$L(s, x, in) = L(s, x, out)$ if s does not refer to x

1. Let all $L(\dots) = \text{false}$ initially
2. Repeat until all statements s satisfy rules 1-4
Pick s where one of 1-4 does not hold and update
using the appropriate rule

Liveness Analysis



- A value can change from false to true, but not the other way around

false < true

- Each value can change only once, so termination is guaranteed
- Once the analysis is computed, it is simple to eliminate dead code

Liveness Analysis

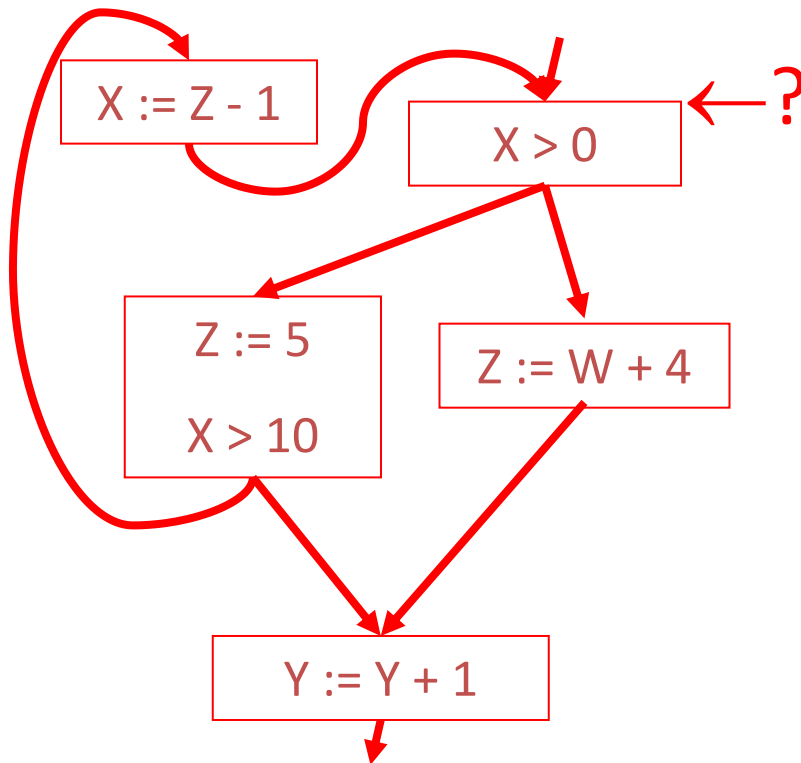
After running the liveness analysis algorithm to completion, which of **W**, **X**, **Y**, and **Z** are live at the program point labeled at right? Assume all variables are dead on exit.

☐ **W**

☐ **X**

☐ **Y**

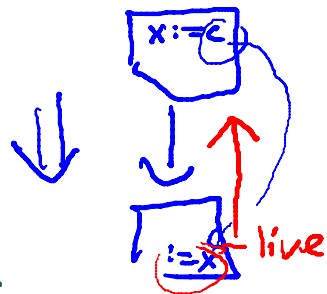
☐ **Z**



We've seen two kinds of analysis:

Constant propagation is a forwards analysis:
information is pushed from inputs to outputs

Liveness is a backwards analysis: information is pushed
from outputs back towards inputs



- There are many other global flow analyses
- Most can be classified as either forward or backward
- Most also follow the methodology of local rules relating information between adjacent program points