Lecture 4: Data-Flow Analysis & Abstract Interpretation Framework

17-355/17-655/17-819: Program Analysis

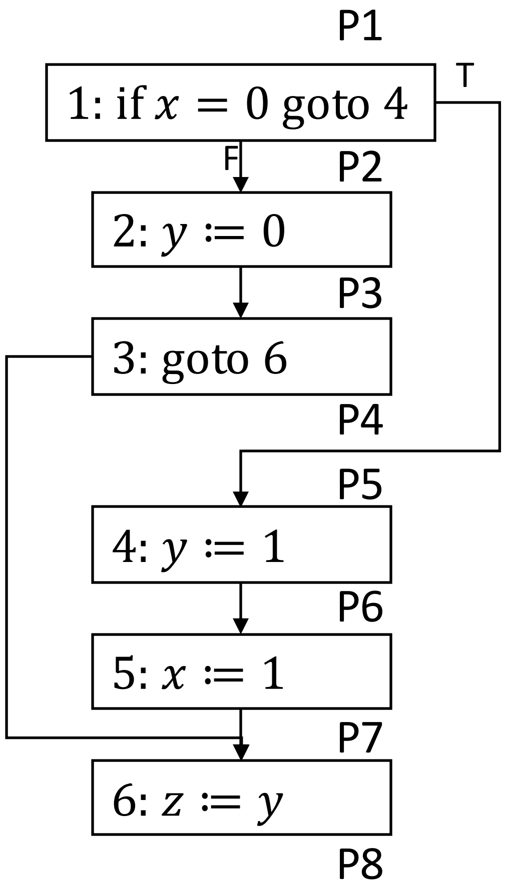
Rohan Padhye

September 9, 2025

* Course materials developed with Jonathan Aldrich Claire Le Goues

Review: Zero Analysis with Branching

$1: \quad \text{if } x = 0 \text{ goto } 4$
$2: \quad y := 0$
$3: \quad \text{goto } 6$
$4: \quad y := 1$
$5: \quad x := 1$
$6: \quad z := y$



	x	y	z
P1	?	?	?
P2	Z_T, N_F	?	?
P3	N	Z	?
P4	N	Z	?
P5	Z	?	?
P6	Z	N	?
P7	N	$\top$	?
P8	N	$\top$	$\top$

Partial Order & Join on set L

$l_1 \sqsubseteq l_2$: l_1 is at least as precise as l_2

reflexive: $\forall l : l \sqsubseteq l$
transitive: $\forall l_1, l_2, l_3 : l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_3 \Rightarrow l_1 \sqsubseteq l_3$
anti-symmetric: $\forall l_1, l_2 : l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_1 \Rightarrow l_1 = l_2$

$l_1 \sqcup l_2$: **join** or *least-upper-bound*… "most precise generalization"

L is a *join-semilattice* iff: $l_1 \sqcup l_2$ always exists and is unique $\forall l_1, l_2 \in L$

$\top$ ("top") is the maximal element

Lattice for Zero Analysis

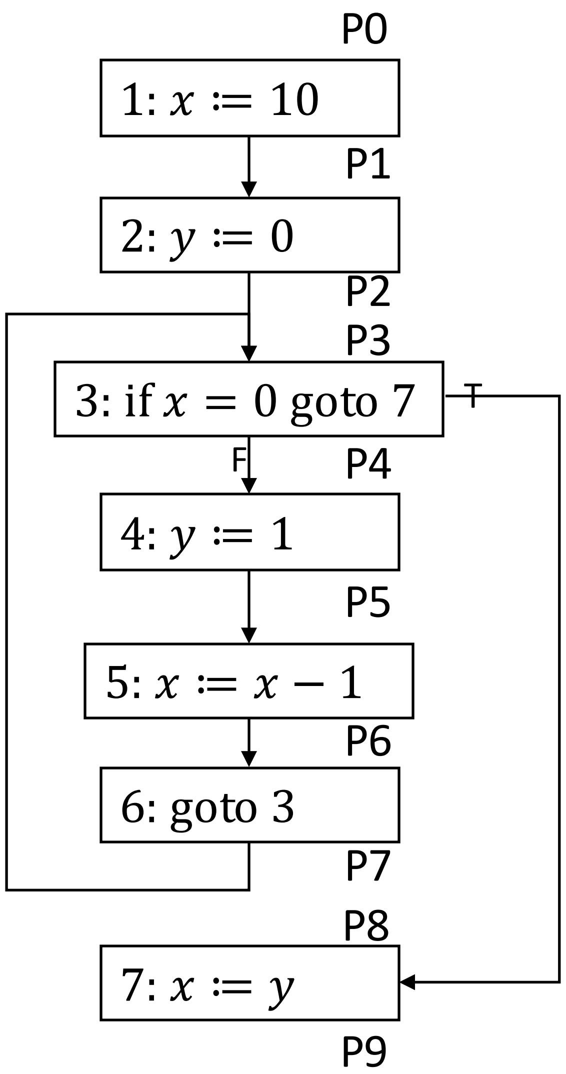
What would this look like?

Data-Flow Analysis

- a lattice $(L, \sqsubseteq)$
- an abstraction function α
- a flow function f
- initial dataflow analysis assumptions, σ_0

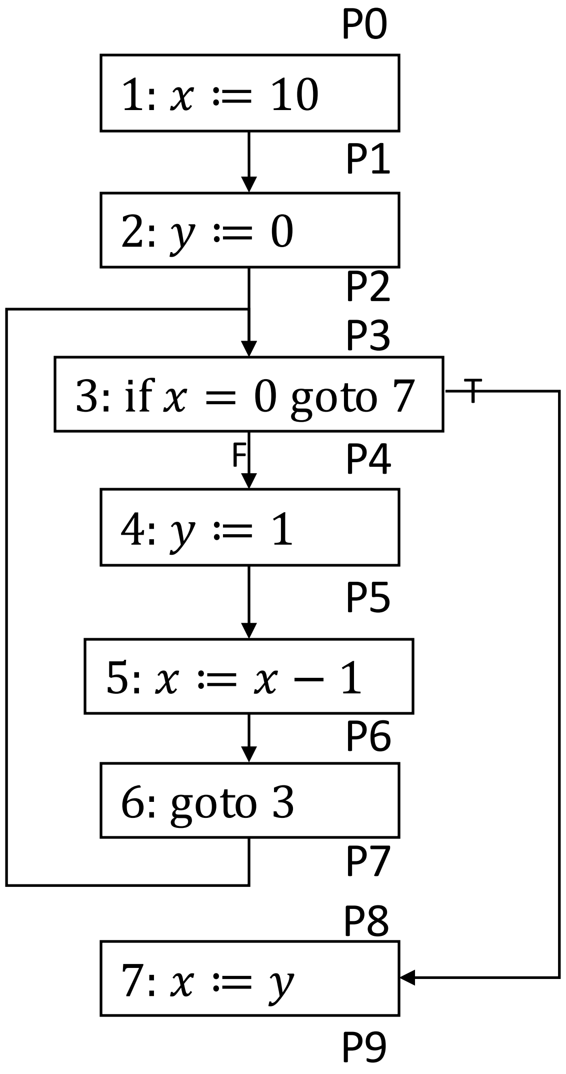
Example of Zero Analysis: Looping Code

$$
\begin{aligned}
1:&\quad x := 10\\
2:&\quad y := 0\\
3:&\quad \text{if } x = 0 \text{ goto } 7\\
4:&\quad y := 1\\
5:&\quad x := x - 1\\
6:&\quad \text{goto } 3\\
7:&\quad x := y
\end{aligned}
$$



Example of Zero Analysis: Looping Code

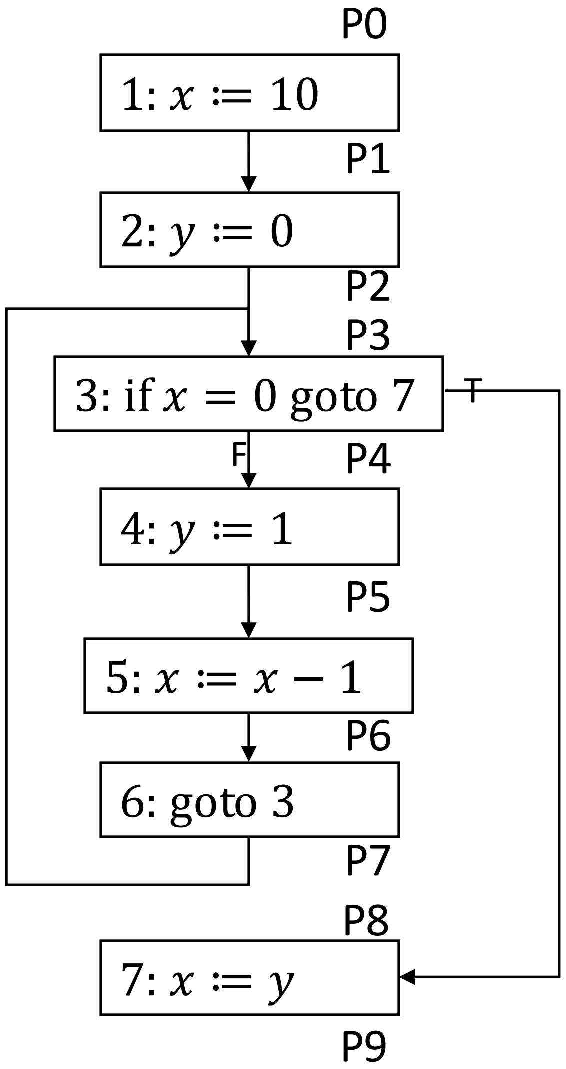
$$
\begin{aligned}
1:&\quad x := 10\\
2:&\quad y := 0\\
3:&\quad \text{if } x = 0 \text{ goto } 7\\
4:&\quad y := 1\\
5:&\quad x := x - 1\\
6:&\quad \text{goto } 3\\
7:&\quad x := y
\end{aligned}
$$



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
P2	N	Z	
P3	N	Z	*first time through...*
P4	N_F	Z	
P5	N	N	
P6	$\top$	N	
P7	$\top$	N	
P8	Z_t	N	*first time through...*
P9	N	N	*first time through...*

Example of Zero Analysis: Looping Code

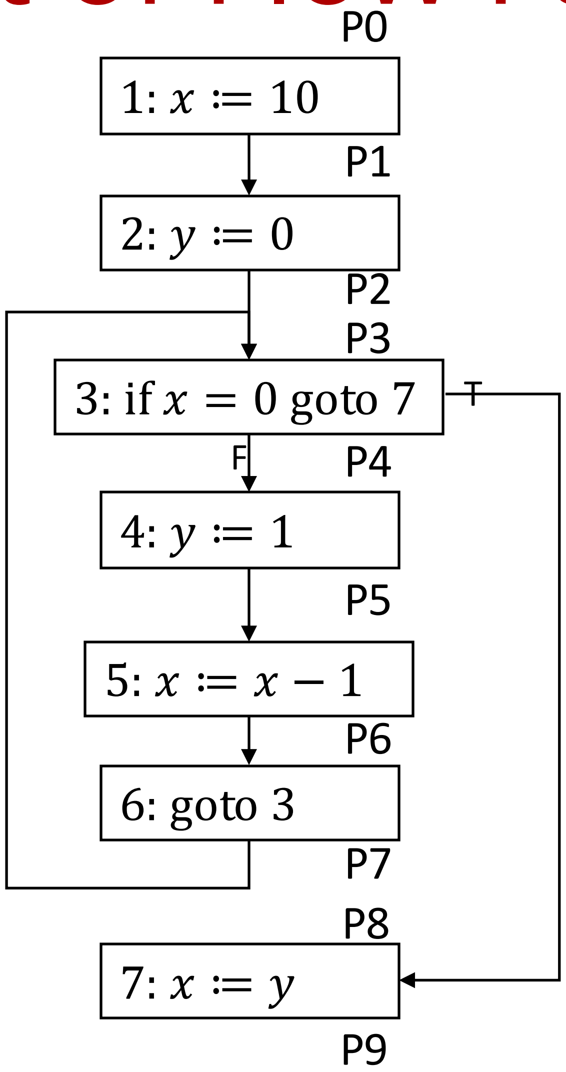
$$
\begin{aligned}
1&: \quad x := 10 \\
2&: \quad y := 0 \\
3&: \quad \text{if } x = 0 \text{ goto } 7 \\
4&: \quad y := 1 \\
5&: \quad x := x - 1 \\
6&: \quad \text{goto } 3 \\
7&: \quad x := y
\end{aligned}
$$



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
P2	N	Z	
P3	$\top$	$\top$	*join*
P4	N_F	$\top$	*updated*
P5	N	N	*already at fixed point*
P6	$\top$	N	*already at fixed point*
P7	$\top$	N	*already at fixed point*
P8	Z_T	$\top$	*updated*
P9	$\top$	$\top$	*updated*

Fixed point of Flow Functions

1 : $x := 10$
2 : $y := 0$
3 : if $x = 0$ goto 7
4 : $y := 1$
5 : $x := x - 1$
6 : goto 3
7 : $x := y$



$$(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) \xrightarrow{f_z} (\sigma'_0, \sigma'_1, \sigma'_2, \dots, \sigma'_n)$$

$$\sigma'_0 = \sigma_0$$

$$\sigma'_1 = f_z[\![x \ := 10]\!](\sigma_0)$$

$$\sigma'_2 = f_z[\![y \ := 0]\!](\sigma_1)$$

$$\sigma'_3 = \sigma_2 \sqcup \sigma_7$$

$$\sigma'_4 = f_z[\![\text{if } x = 10 \text{ goto } 7]\!]_F(\sigma_3)$$

$$\vdots$$

$$\sigma'_8 = f_z[\![\text{if } x = 10 \text{ goto } 7]\!]_T(\sigma_3)$$

$$\sigma'_9 = f_z[\![x \ := y]\!](\sigma_8)$$

Fixed point of Flow Functions

$$(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) \xrightarrow{f_z} (\sigma'_0, \sigma'_1, \sigma'_2, \dots, \sigma'_n)$$

$$\sigma'_0 = \sigma_0$$

$$\sigma'_1 = f_z[\![x := 10]\!](\sigma_0)$$

$$\sigma'_2 = f_z[\![y := 0]\!](\sigma_1)$$

$$\sigma'_3 = \sigma_2 \sqcup \sigma_7$$

$$\sigma'_4 = f_z[\![\text{if } x = 10 \text{ goto } 7]\!]_F(\sigma_3)$$

$$\vdots$$

$$\sigma'_8 = f_z[\![\text{if } x = 10 \text{ goto } 7]\!]_T(\sigma_3)$$

$$\sigma'_9 = f_z[\![x := y]\!](\sigma_8)$$

Fixed point!

$$(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) = f_z(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n)$$

Correctness theorem:
If data-flow analysis is well designed*, then any fixed point of the analysis is sound.

* we will define these properties and prove this theorem in two weeks!

More on joins and lattices

$$(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) \xrightarrow{f_z} (\sigma'_0, \sigma'_1, \sigma'_2, \dots, \sigma'_n)$$

Hold up! How do you

$$\sigma'_0 = \sigma_0$$

$$\sigma'_1 = f_z[\![x := 10]\!](\sigma_0)$$

$$\sigma'_2 = f_z[\![y := 0]\!](\sigma_1)$$

$$\sigma'_3 = \sigma_2 \sqcup \sigma_7$$

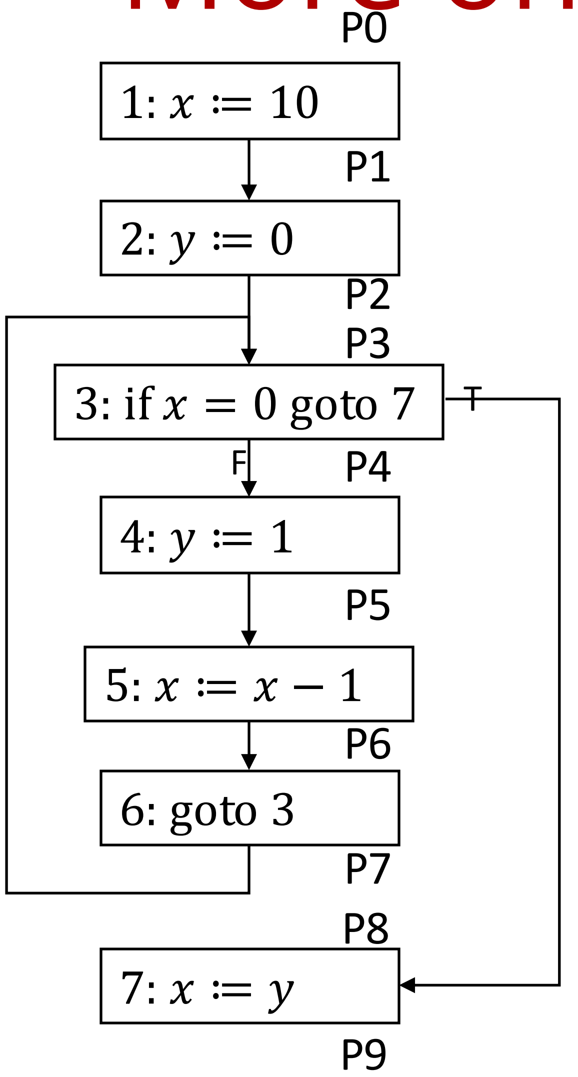
$$\sigma'_4 = f_z[\![\text{if } x = 10 \text{ goto } 7]\!]_F(\sigma_3)$$

$$\vdots$$

$$\sigma'_8 = f_z[\![\text{if } x = 10 \text{ goto } 7]\!]_T(\sigma_3)$$

$$\sigma'_9 = f_z[\![x := y]\!](\sigma_8)$$

More on joins and lattices



	x	y	
P0	⊤	⊤	
P1	N	⊤	
P2	N	Z	
P3	N	Z	*first time through...*
P4			
P5			
P6			
P7			
P8			
P9			

$\sigma'_3 = \sigma_2 \sqcup \sigma_7$

What should be the initial value for σ_7 ????

More on joins and lattices

Enter: ⊥ (“bottom”)

What would the **complete lattice** for Zero Analysis look like?

for all $l \in L$:

$\bot \sqsubseteq l$ $\quad$ $l \sqsubseteq \top$

$\bot \sqcup l = l$ $\quad$ $l \sqcup \top = \top$

A lattice with both ⊥ and ⊤ defined is called a ***Complete Lattice***

More on joins and lattices

$\sigma : Var \rightarrow L$ where $L = \{Z, N, \bot, \top\}$ and $Var = \{x, y\}$

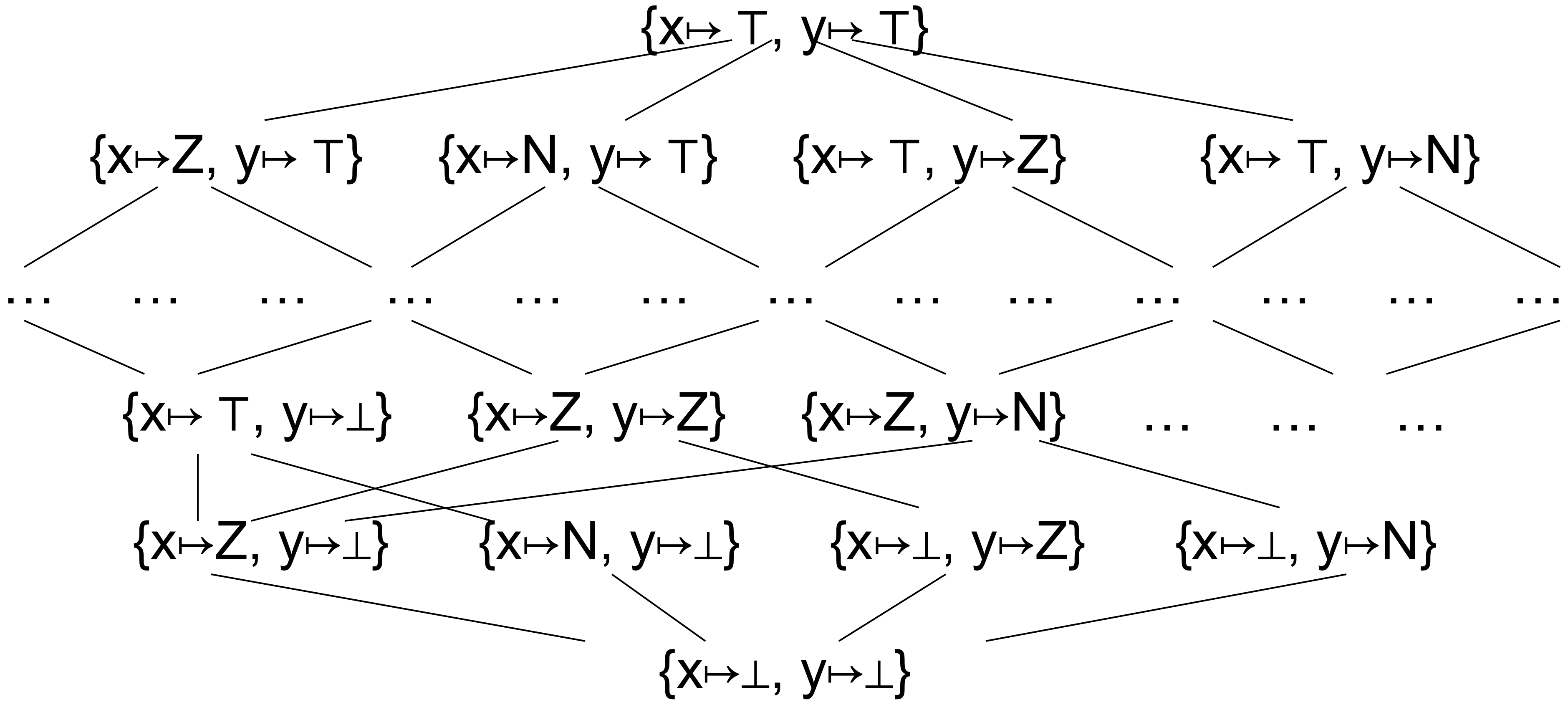
$$\sigma_1 \sqcup \sigma_2 = \{\, x \mapsto \sigma_1(x) \sqcup \sigma_2(x), \quad y \mapsto \sigma_1(y) \sqcup \sigma_2(y) \,\}$$

Exercise: Define lifted $\sqsubseteq$ in terms of ordering on L

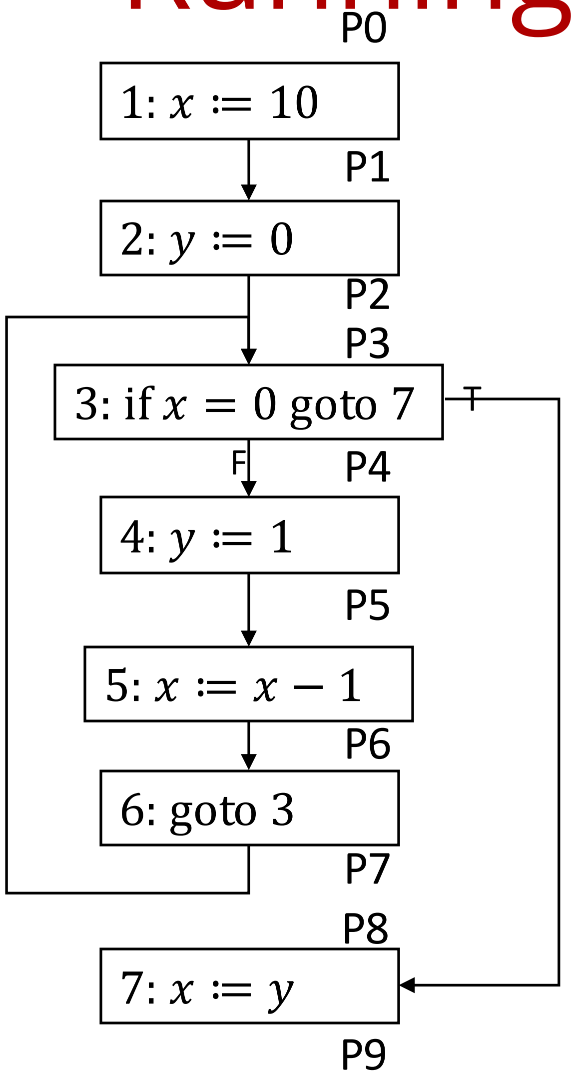
$\sigma_1 \sqsubseteq \sigma_2 =$ **???**

More on joins and lattices

Lifting a complete lattice gives another complete lattice

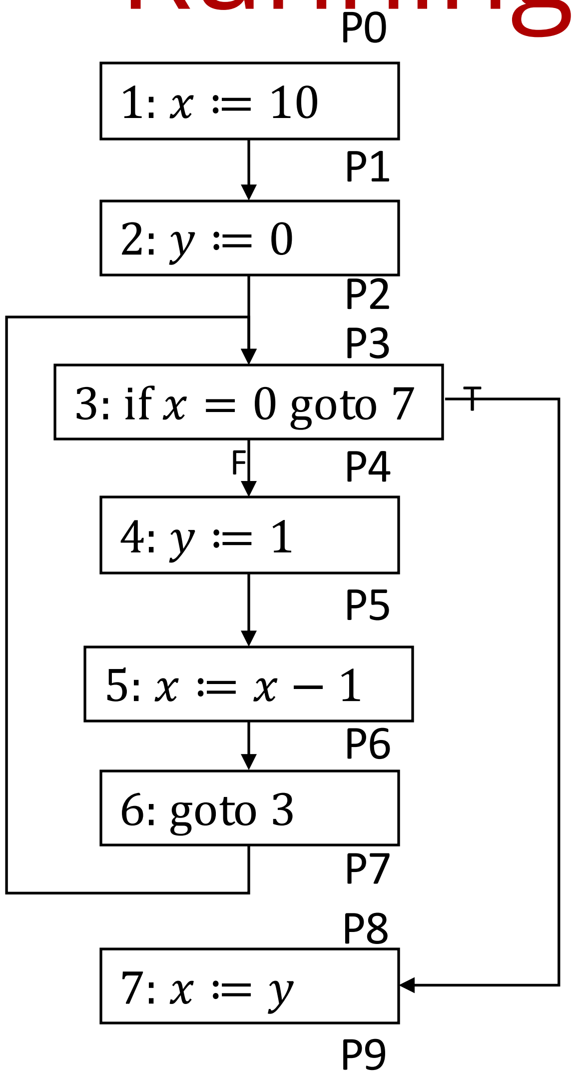


Running a Data Flow Analysis



	x	y
P0	$\top$	$\top$
P1	$\bot$	$\bot$
P2	$\bot$	$\bot$
P3	$\bot$	$\bot$
P4	$\bot$	$\bot$
P5	$\bot$	$\bot$
P6	$\bot$	$\bot$
P7	$\bot$	$\bot$
P8	$\bot$	$\bot$
P9	$\bot$	$\bot$

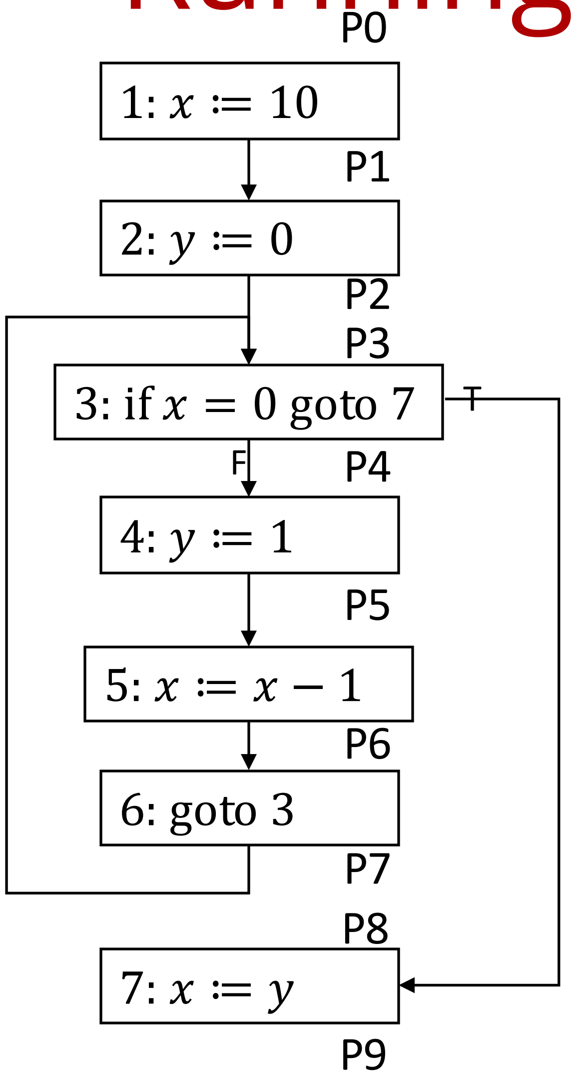
Running a Data Flow Analysis



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
P2	N	Z	
P3	N	Z	*first time through...*
P4	$\bot$	$\bot$	
P5	$\bot$	$\bot$	
P6	$\bot$	$\bot$	
P7	$\bot$	$\bot$	
P8	$\bot$	$\bot$	
P9	$\bot$	$\bot$	

$\sigma'_3 = \sigma_2 \sqcup \sigma_7$

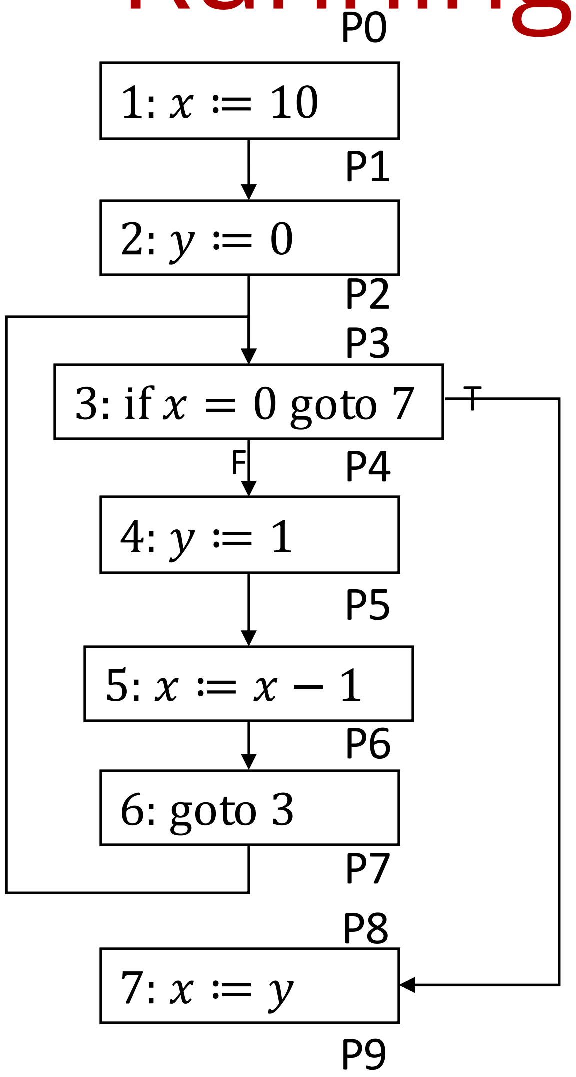
Running a Data Flow Analysis



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
P2	N	Z	
P3	N	Z	*first time through...*
P4	N_F	Z	
P5	N	N	
P6	$\top$	N	
P7	$\top$	N	
P8	Z_t	N	*first time through...*
P9	N	N	*first time through...*

$\sigma'_3 = \sigma_2 \sqcup \sigma_7$

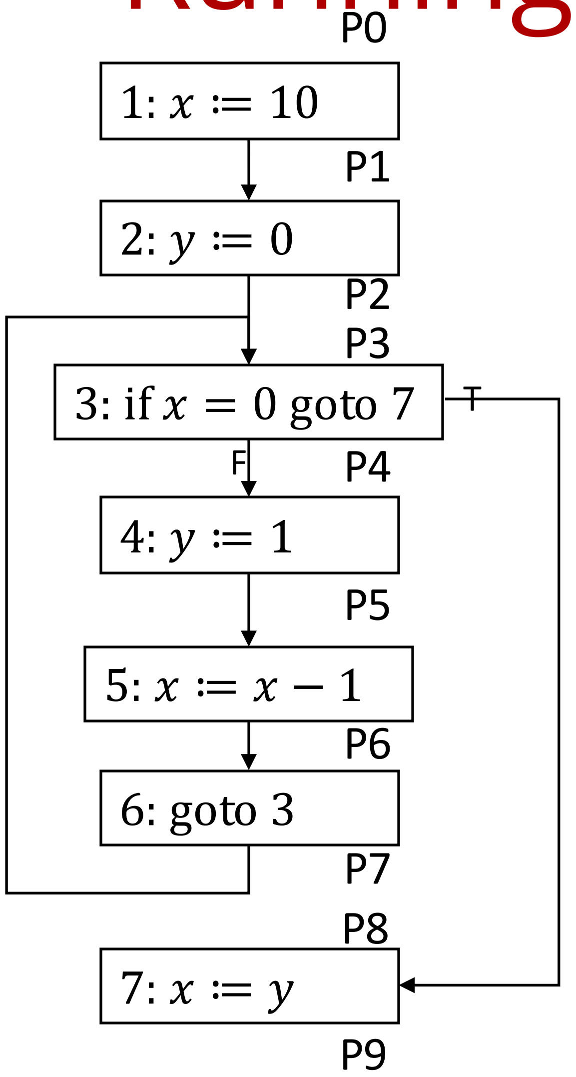
Running a Data Flow Analysis



	x	y	
P0	⊤	⊤	
P1	N	⊤	
P2	N	Z	
P3	⊤	⊤	*join*
P4	N_F	Z	
P5	N	N	
P6	⊤	N	
P7	⊤	N	
P8	Z_t	N	*first time through...*
P9	N	N	*first time through...*

$\sigma'_3 = \sigma_2 \sqcup \sigma_7$

Running a Data Flow Analysis



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
P2	N	Z	
P3	$\top$	$\top$	*join*
P4	N_F	$\top$	*updated*
P5	N	N	*already at fixed point*
P6	$\top$	N	*already at fixed point*
P7	$\top$	N	*already at fixed point*
P8	Z_T	$\top$	*updated*
P9	$\top$	$\top$	*updated*

What’s the Algorithm?

Analysis Execution Strategy

```
for Node n in cfg
    input[n] = ⊥
input[0] = initialDataflowInformation

while not at fixed point
    pick a node n in program
    output = flow(n, input[n])
    for Node j in sucessors(n)
        input[j] = input[j] ⊔ output
```

Kildall's Algorithm

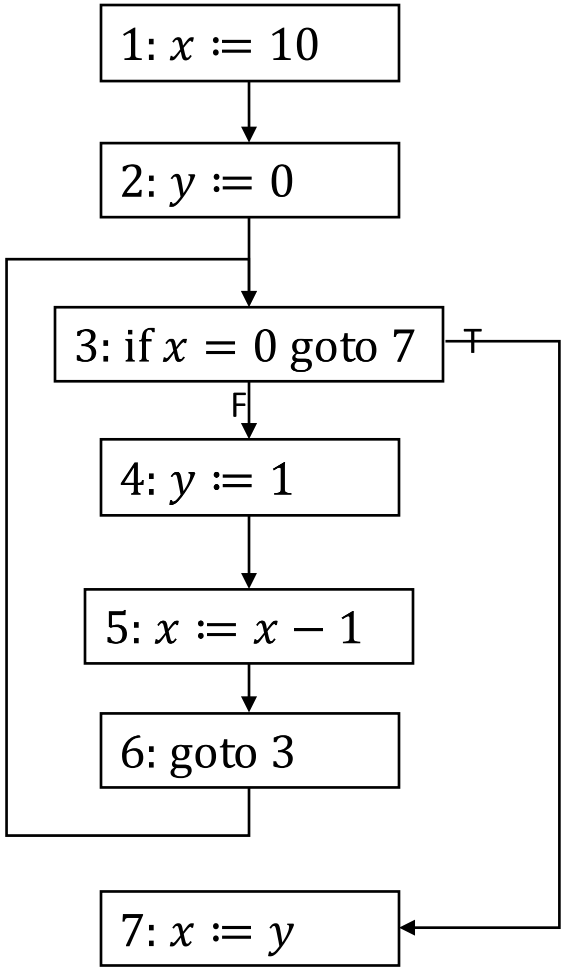
```
worklist = ∅
for  Node n in cfg
    input[n] = output[n] = ⊥
    add n to worklist
input[0] = initialDataflowInformation

while worklist is not empty
    take a Node n off the worklist
    output[n] = flow(n, input[n])
    for Node j in succs(n)
          newInput = input[j] ⊔ output[n]
          if newInput ≠ input[j]
                input[j] = newInput
                add j to worklist
```

What order to process worklist nodes in?

- Random? Queue? Stack?
- Any order is valid (!!)
- Some orders are better in practice
  - Topological sorts are nice
  - Explore loops inside out
  - Reverse postorder!

Exercise: Apply Kildall's Worklist Algorithm for Zero Analysis



Performance of Kildall's Algorithm

- Why is it guaranteed to terminate?
- What is its complexity?