Lecture 4: Data-Flow Analysis (contd.)

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

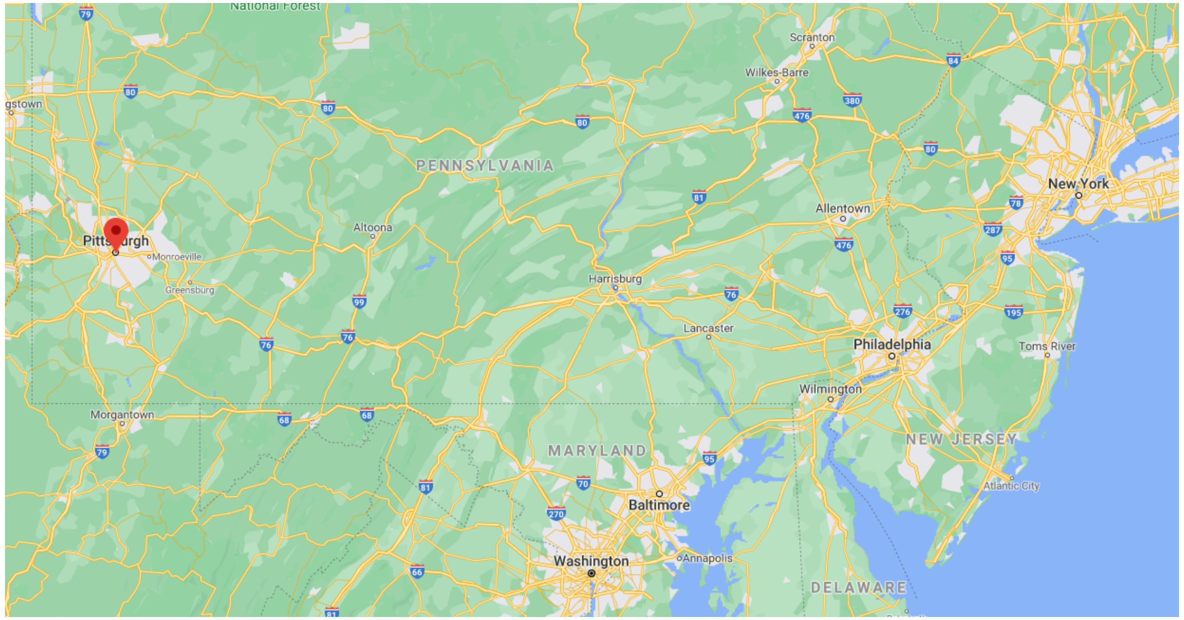
Rohan Padhye and Jonathan Aldrich

February 11, 2021

* Course materials developed with Claire Le Goues

Random Facts #1

"You are here" maps don't lie



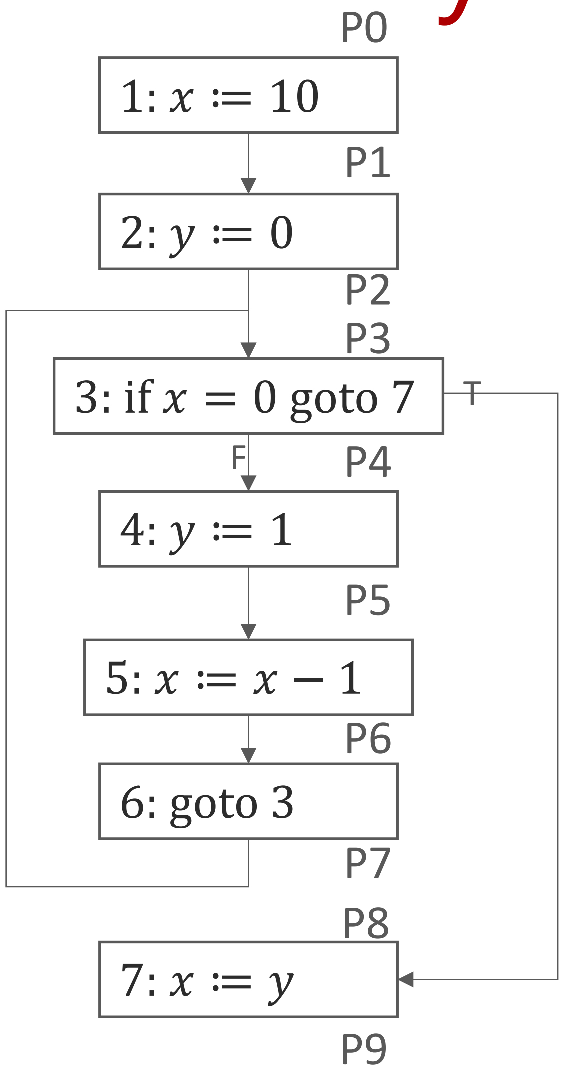
Python 3.8:

```
exec(s:='print("exec(s:=%r)"%s)')
```

What mathematical concept is common to both these facts?

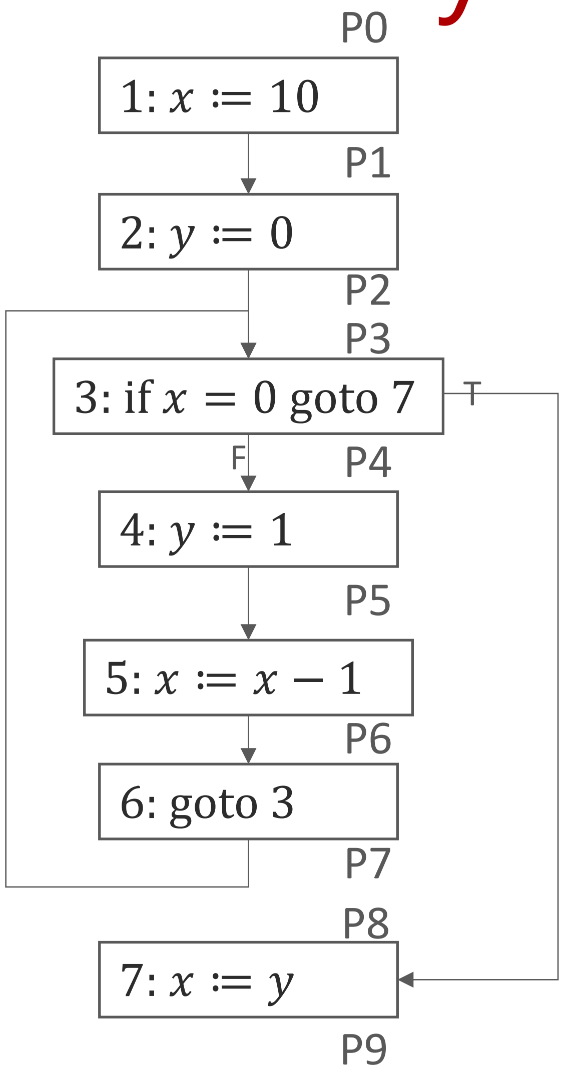
Example of Zero Analysis: Looping Code

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



Example of Zero Analysis: Looping Code

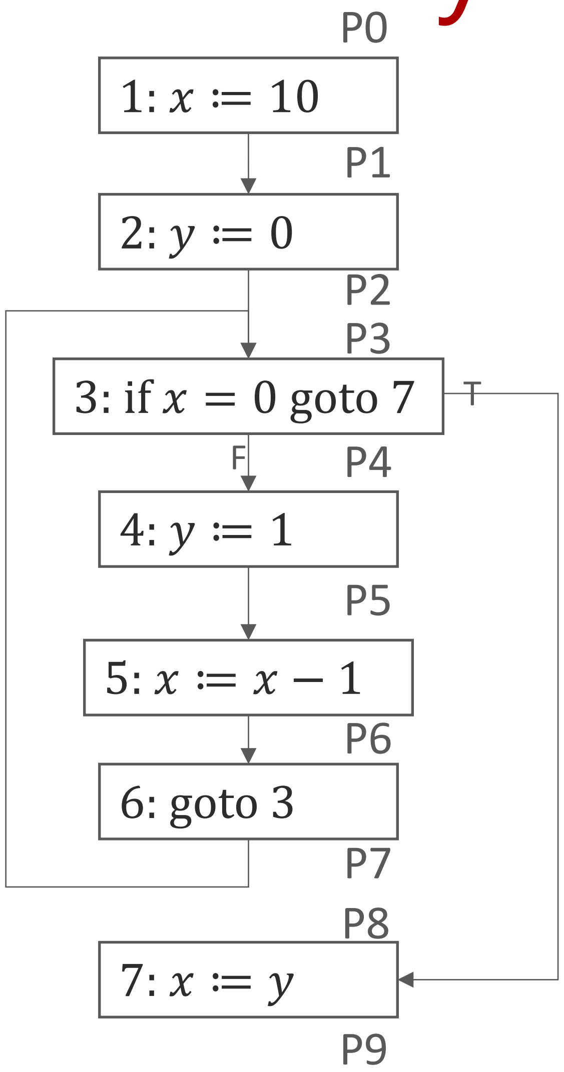
$$
\begin{array}{ll}
1: & x := 10 \\
2: & y := 0 \\
3: & \text{if } x = 0 \text{ goto } 7 \\
4: & y := 1 \\
5: & x := x - 1 \\
6: & \text{goto } 3 \\
7: & x := y
\end{array}
$$



	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

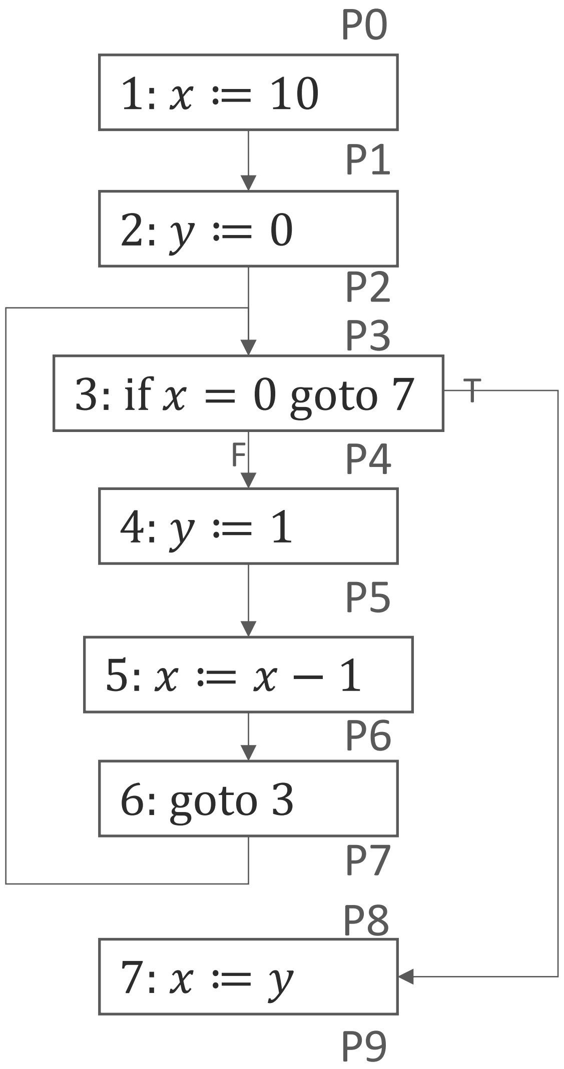
$$
\begin{array}{ll}
1: & x := 10 \\
2: & y := 0 \\
3: & \text{if } x = 0 \text{ goto } 7 \\
4: & y := 1 \\
5: & x := x - 1 \\
6: & \text{goto } 3 \\
7: & x := y
\end{array}
$$



	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

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



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

$$\sigma'_0 = \sigma_0$$

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

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

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

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

$$\vdots$$

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

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

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!

$$\boxed{(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) = f_Z(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n)}$$

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

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

Hold up! How do you

$$\sigma'_0 = \sigma_0$$

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

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

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

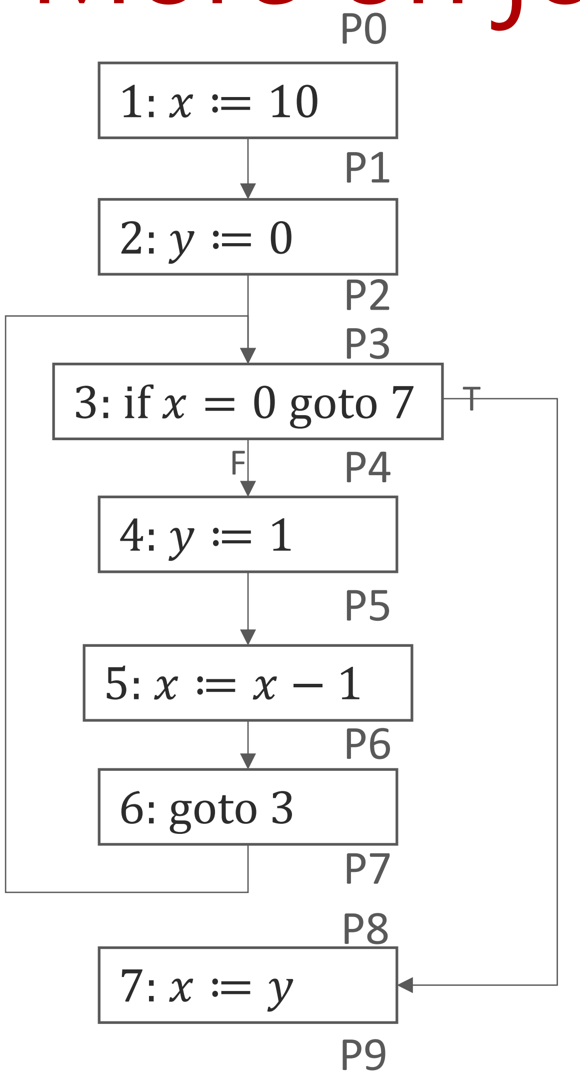
$$\sigma'_4 = f_Z[\![\text{if } x = 10 \text{ goto } 7]\!]_F(\sigma_3)$$

$$\vdots$$

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

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

More on joins and lattices



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
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 $l \in L$:

$\bot \sqsubseteq l$ $\qquad$ $l \sqsubseteq \top$

$\bot \sqcup l = l$ $\qquad$ $l \sqcup \top = \top$

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

More on joins and lattices

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

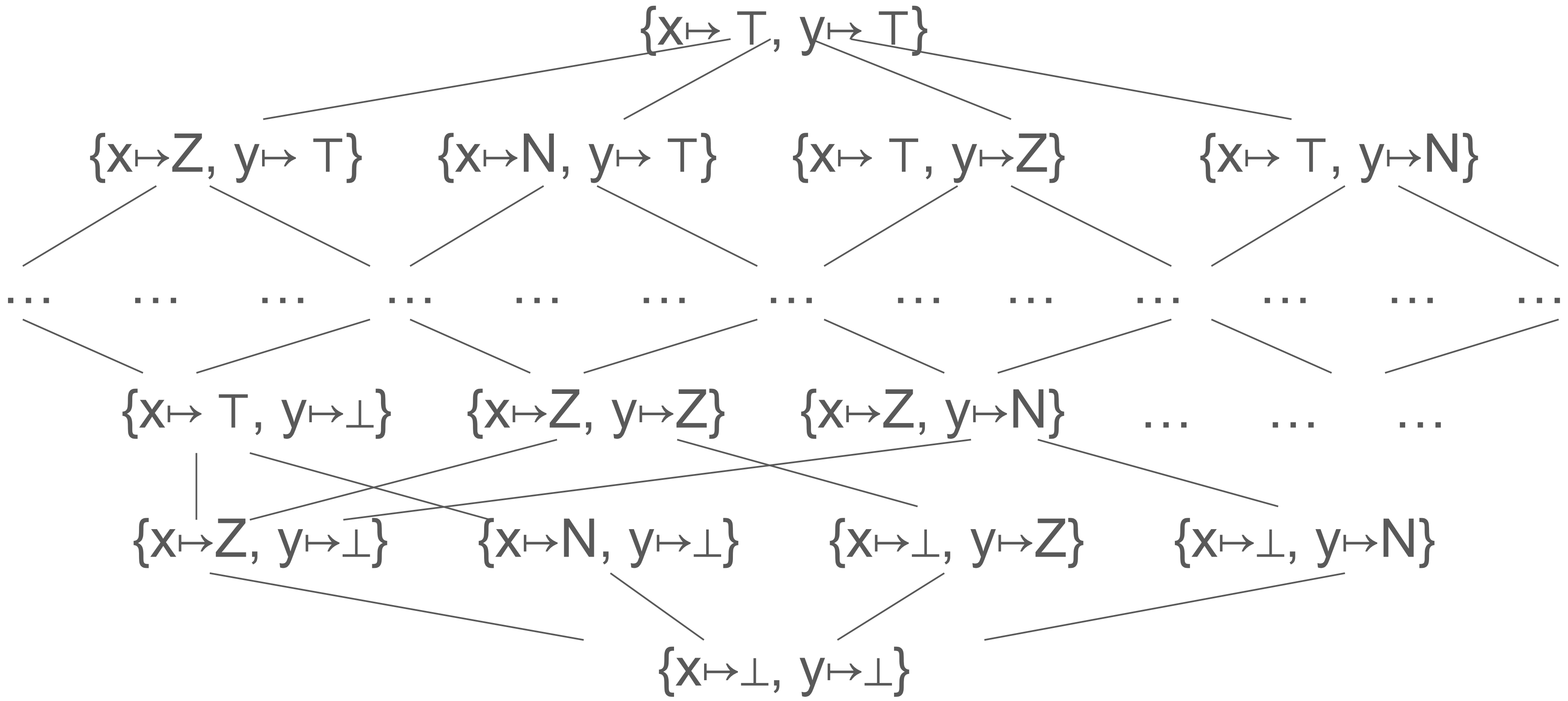
$\sigma_1 \sqcup \sigma_2 = \{\, x \mapsto \sigma_1(x) \sqcup \sigma_2(x), \qquad y \mapsto \sigma_1(y) \sqcup \sigma_2(y) \,\}$

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

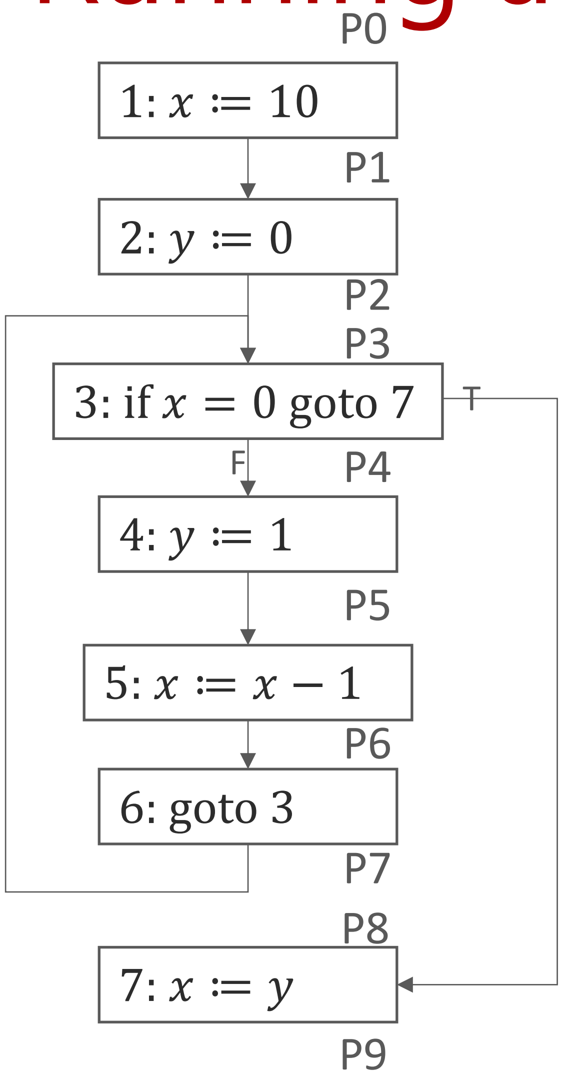
$\sigma_1 \sqsubseteq \sigma_2 =$ **???**

More on joins and lattices

Lifting a complete lattice gives another complete lattice

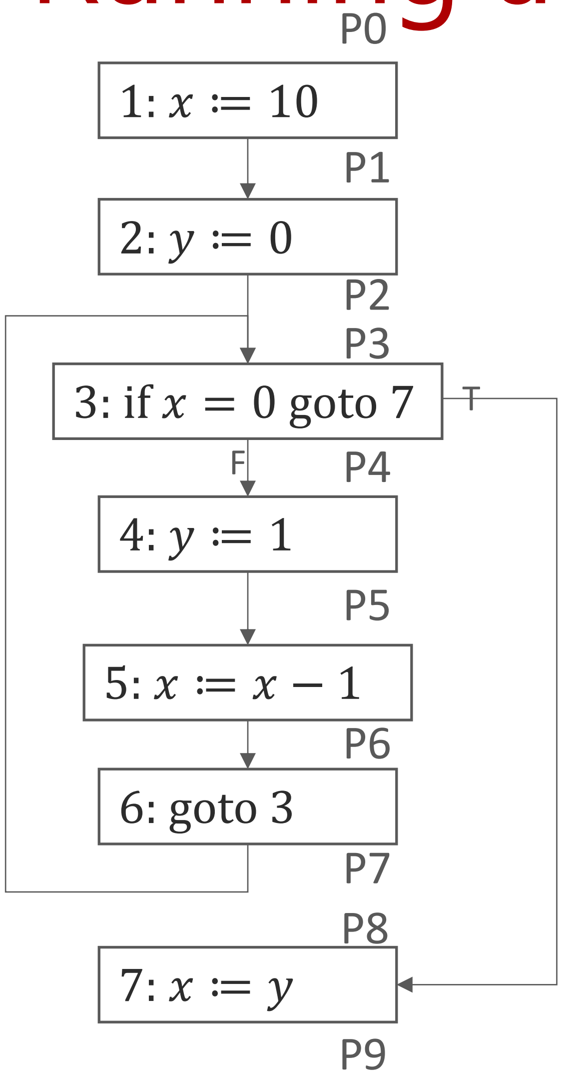


Running a Data Flow Analysis



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

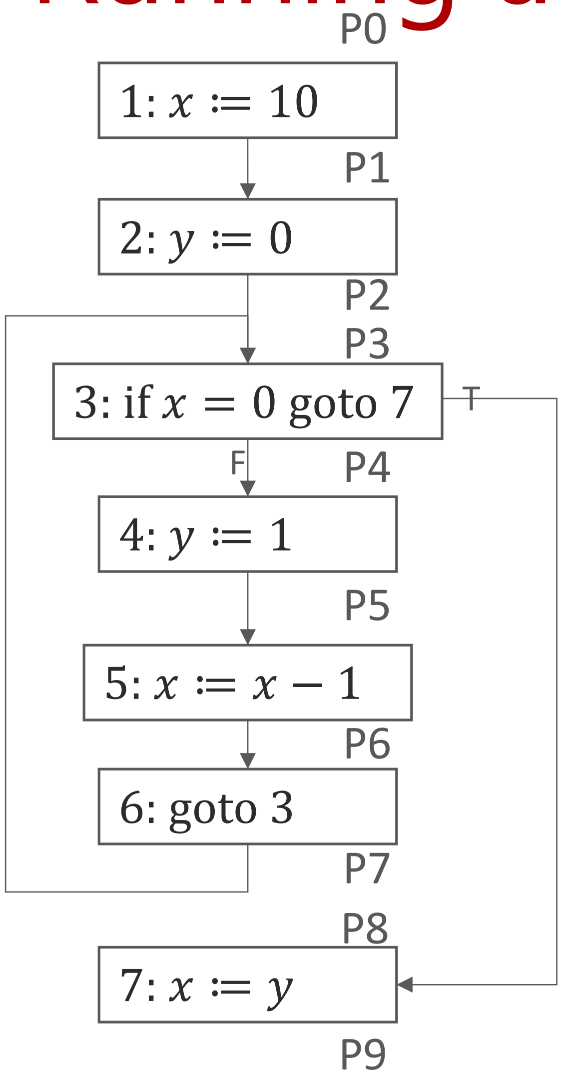
Running a Data Flow Analysis



	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	$\bot$	$\bot$	

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

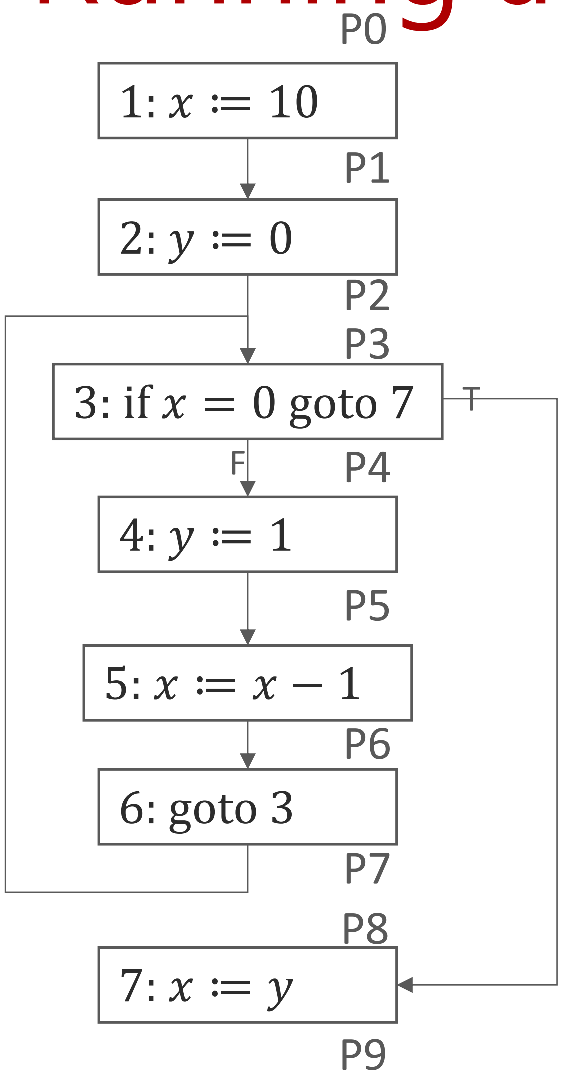
Running a Data Flow Analysis



	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...*

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

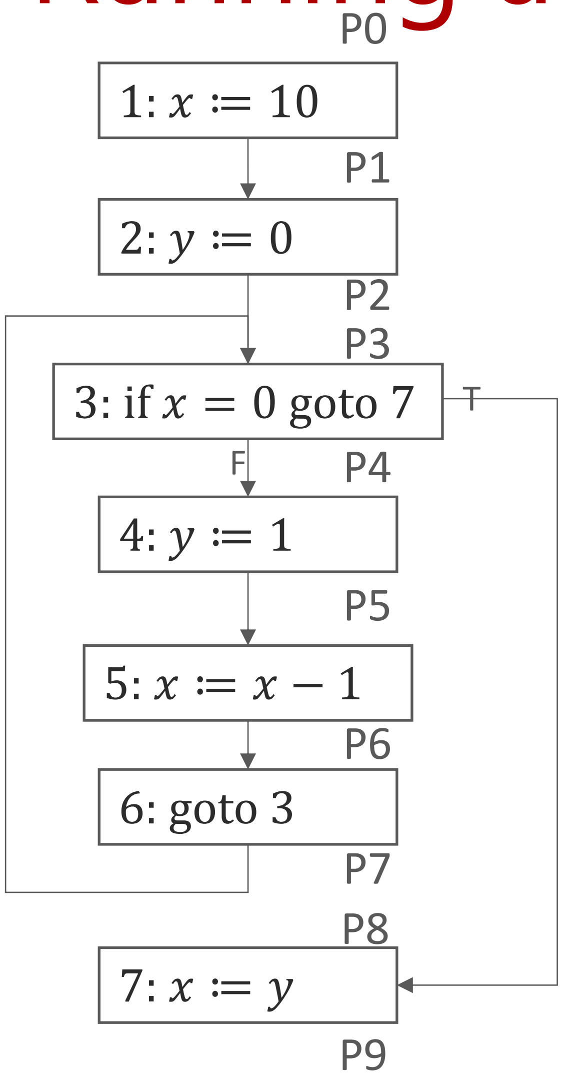
Running a Data Flow Analysis



	x	y	
P0	$\top$	$\top$	
P1	N	$\top$	
P2	N	Z	
P3	$\top$	$\top$	*join*
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	$\top$	$\top$	
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

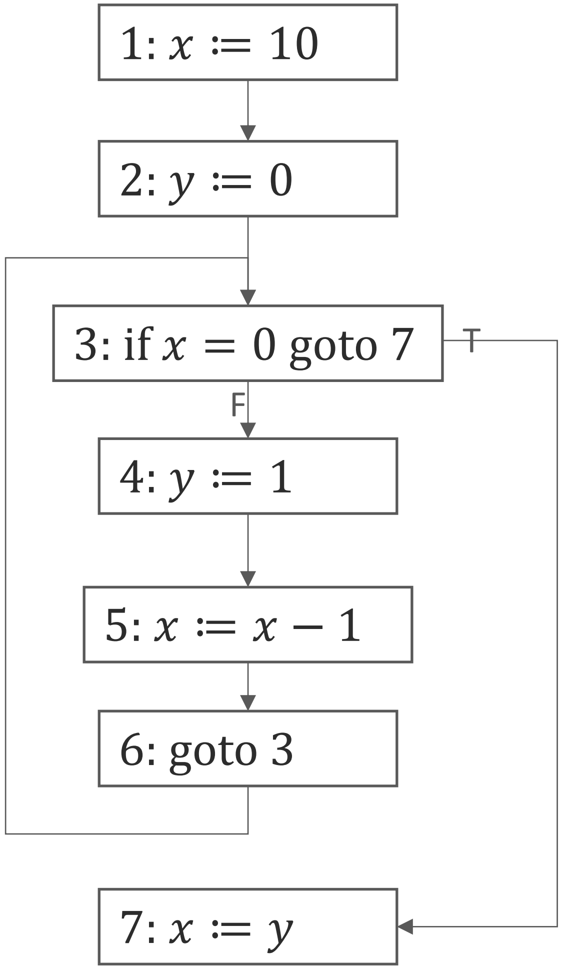
```
worklist = ∅
for  Node n in cfg
     input[n] = output[n] = ⊥
     add n to worklist
input[0] = initialDataflowInformation

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?