Lecture 3b: Data-Flow Analysis

17-355/17-665/17-819: Program Analysis

Rohan Padhye

Jan 25, 2022

* Course materials developed with Jonathan Aldrich and Claire Le Goues

Data-Flow Analysis

Computes universal properties about program state at specific program points. (e.g. will *x* be zero at line 7?)

- About program state
  - About data store (e.g. variables, heap memory)
  - Not about control (e.g. termination, performance)
- At program points
  - Statically identifiable (e.g. line 7, or when foo() calls bar())
  - Not dynamically computed (E.g. when x is 12 or when foo() is invoked 12 times)
- Universal
  - Reasons about all possible executions (always/never/maybe)
  - Not about specific program paths (see: symbolic execution, testing)

Abstraction

$$\sigma \in Var \to L$$

$$\alpha : \mathbb{Z} \to L$$

Abstraction

$\sigma \in Var \rightarrow L$

$\alpha : \mathbb{Z} \rightarrow L$

Zero Analysis

$L = \{Z, N, \top\}$

$$\alpha_Z(0) = Z$$
$$\alpha_Z(n) = N \textbf{where } n \neq 0$$

Flow Functions for Zero Analysis

A flow function maps values from σ to σ

$f[\![I]\!]$ -- flow across instruction I (think: "abstract semantics")

$$f_Z[\![x := 0]\!](\sigma) =$$

$$f_Z[\![x := n]\!](\sigma) =$$

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

$$f_Z[\![x := y \; op \; z]\!](\sigma) =$$

$$f_Z[\![\text{goto } n]\!](\sigma) =$$

$$f_Z[\![\text{if } x = 0 \text{ goto } n]\!](\sigma) =$$

Flow Functions for Zero Analysis

A flow function maps values from σ to σ

$f[\![I]\!]$ -- flow across instruction I (think: “abstract semantics”)

$$f_Z[\![x := 0]\!](\sigma) = \sigma[x \mapsto Z]$$

$$f_Z[\![x := n]\!](\sigma) = \sigma[x \mapsto N] \text{ where } n \neq 0$$

$$f_Z[\![x := y]\!](\sigma) = \sigma[x \mapsto \sigma(y)]$$

$$f_Z[\![x := y\ op\ z]\!](\sigma) = \sigma[x \mapsto \top]$$

$$f_Z[\![\text{goto } n]\!](\sigma) = \sigma$$

$$f_Z[\![\text{if } x = 0 \text{ goto } n]\!](\sigma) = \sigma$$

Flow Functions for Zero Analysis

Specializing for Precision

$$f_Z[\![x := y - y]\!](\sigma) \quad =$$

$$f_Z[\![x := y + z]\!](\sigma) \quad =$$

Flow Functions for Zero Analysis

Specializing for Precision

$$f_Z[\![x := y - y]\!](\sigma) = \sigma[x \mapsto Z]$$

$$f_Z[\![x := y + z]\!](\sigma) = \sigma[x \mapsto \sigma(y)] \quad \text{where } \sigma(z) = Z$$

Exercise: Define another flow function for some arithmetic instruction and certain conditions where you can also provide a more precise result than ⊤

Flow Functions for Zero Analysis

Specializing for Precision

$$
\begin{aligned}
f_Z[\![\text{if } x = 0 \text{ goto } n]\!]_T(\sigma) &= \\
f_Z[\![\text{if } x = 0 \text{ goto } n]\!]_F(\sigma) &=
\end{aligned}
$$

Flow Functions for Zero Analysis

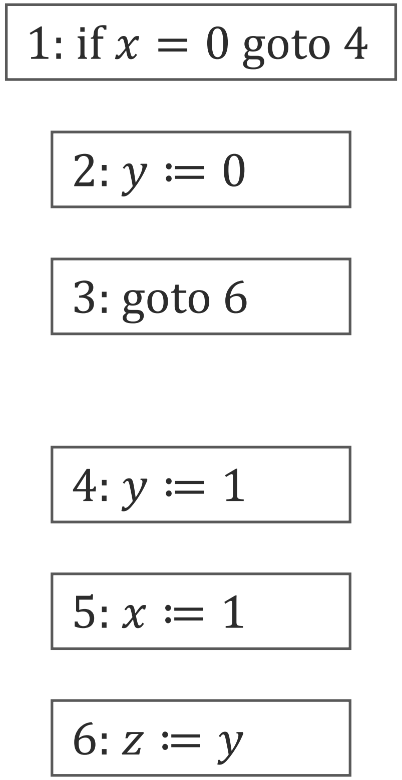
Specializing for Precision

$$
\begin{aligned}
f_Z[\![\text{if } x = 0 \text{ goto } n]\!]_T(\sigma) &= \sigma[x \mapsto Z] \\
f_Z[\![\text{if } x = 0 \text{ goto } n]\!]_F(\sigma) &= \sigma[x \mapsto N]
\end{aligned}
$$

Exercise: Define a flow function for a conditional branch testing whether a variable x < 0

Control-flow Graphs

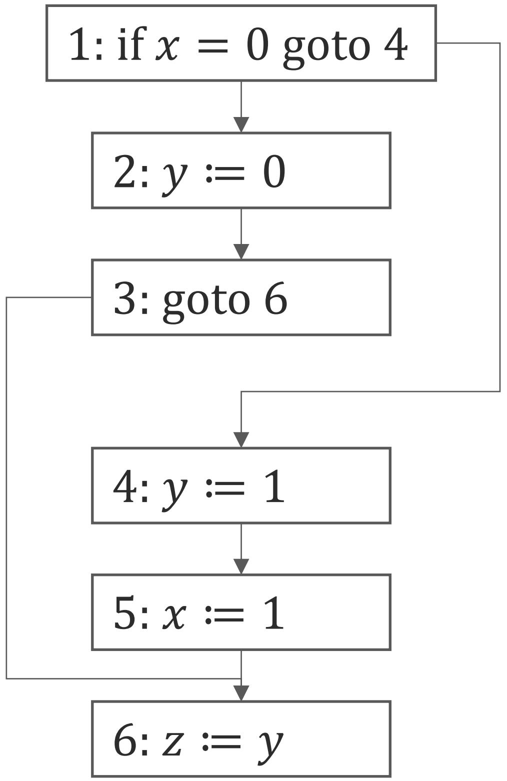
$1:\quad \text{if } x = 0 \text{ goto } 4$
$2:\quad y := 0$
$3:\quad \text{goto } 6$
$4:\quad y := 1$
$5:\quad x := 1$
$6:\quad z := y$



Nodes = Statements
Edges = (s1, s2) is an edge iff s1 and s2 can be executed consecutively aka ”control flow”

Control-flow Graphs

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



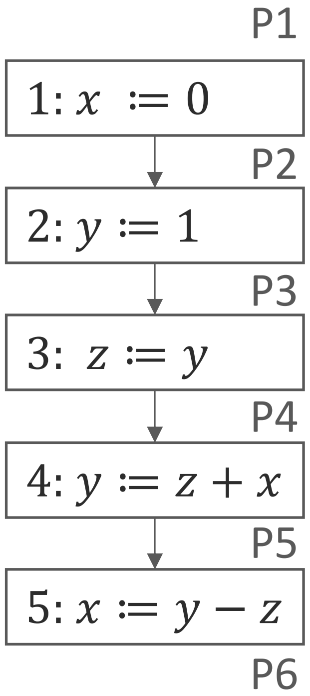
Nodes = Statements
Edges = (s1, s2) is an edge iff s1 and s2 can be executed consecutively aka "control flow"

Common properties of CFGs:

- Weakly connected
- Only one entry node
- Only one exit (terminal) node

Example of Zero Analysis: Straightline Code

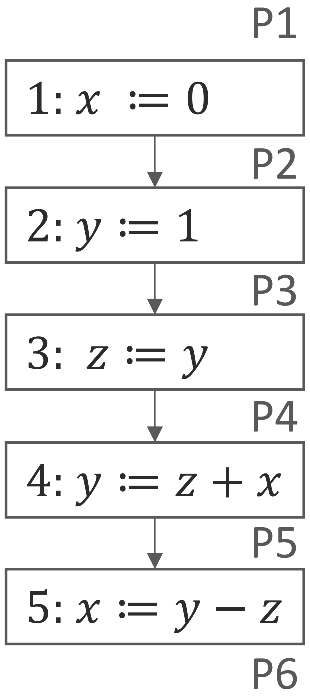
$$
\begin{aligned}
1:&\quad x := 0\\
2:&\quad y := 1\\
3:&\quad z := y\\
4:&\quad y := z + x\\
5:&\quad x := y - z
\end{aligned}
$$



	x	y	z
P1			
P2			
P3			
P4			
P5			
P6			

Example of Zero Analysis: Straightline Code

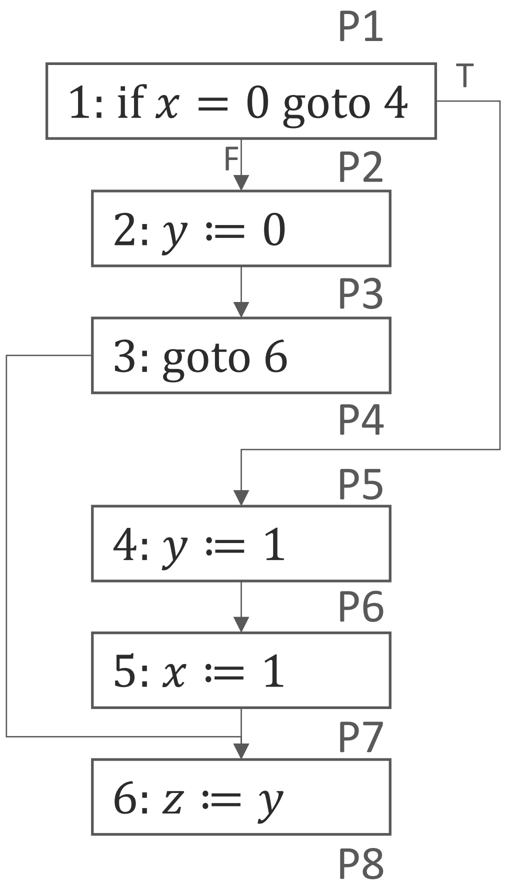
$$
\begin{aligned}
1:&\quad x := 0\\
2:&\quad y := 1\\
3:&\quad z := y\\
4:&\quad y := z + x\\
5:&\quad x := y - z
\end{aligned}
$$



	x	y	z
P1	?	?	?
P2	Z	?	?
P3	Z	N	?
P4	Z	N	N
P5	Z	N	N
P6	$\top$	N	N

Example of Zero Analysis: Branching Code

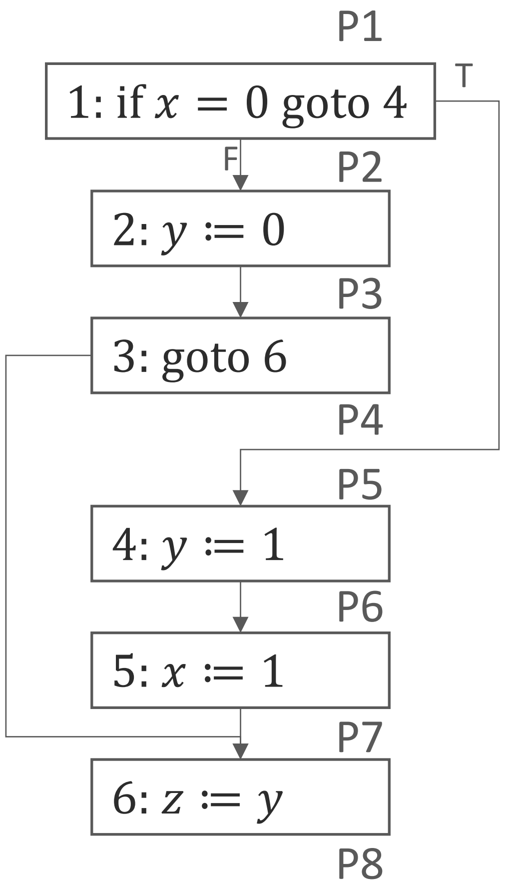
$$
\begin{aligned}
1 &: \quad \text{if } x = 0 \text{ goto } 4 \\
2 &: \quad y := 0 \\
3 &: \quad \text{goto } 6 \\
4 &: \quad y := 1 \\
5 &: \quad x := 1 \\
6 &: \quad z := y
\end{aligned}
$$



	x	y	z
P1			
P2			
P3			
P4			
P5			
P6			
P7			
P8			

Example of Zero Analysis: Branching Code

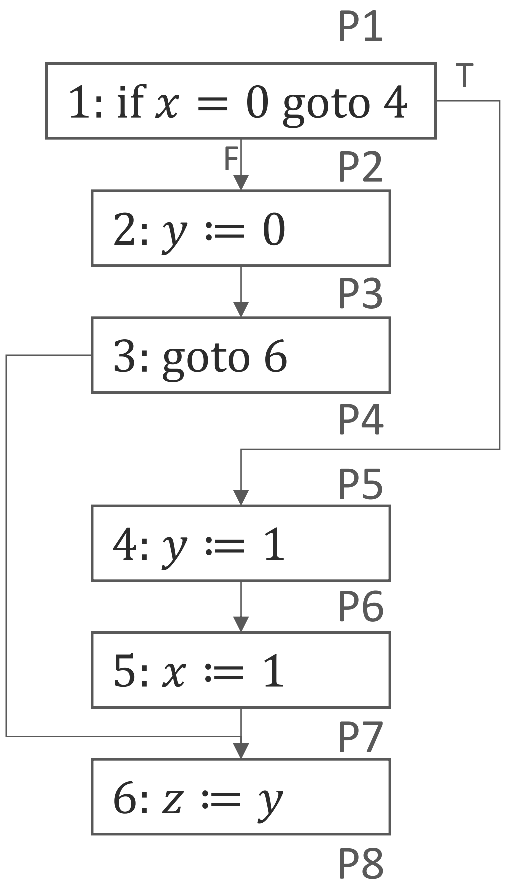
```
1 :  if x = 0 goto 4
2 :  y := 0
3 :  goto 6
4 :  y := 1
5 :  x := 1
6 :  z := y
```



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

Example of Zero Analysis: Branching Code

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



	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	$\top$	$\top$

Next Time

- Lattices
- Definition of a Data-Flow Analysis
- Solution of a Data-Flow Analysis
- Kildall's Algorithm