Live Variables Analysis

Uday Khedker

(www.cse.iitb.ac.in/~uday)

Department of Computer Science and Engineering,

Indian Institute of Technology, Bombay

Dec 2019

Part 1

About These Slides

Copyright

These slides constitute the lecture notes for CS618 Program Analysis course at IIT Bombay and have been made available as teaching material accompanying the book:

- Uday Khedker, Amitabha Sanyal, and Bageshri Karkare. *Data Flow Analysis: Theory and Practice*. CRC Press (Taylor and Francis Group). 2009.

  (Indian edition published by Ane Books in 2013)

Apart from the above book, some slides are based on the material from the following books

- M. S. Hecht. *Flow Analysis of Computer Programs*. Elsevier North-Holland Inc. 1977.
- F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag. 1998.

Outline

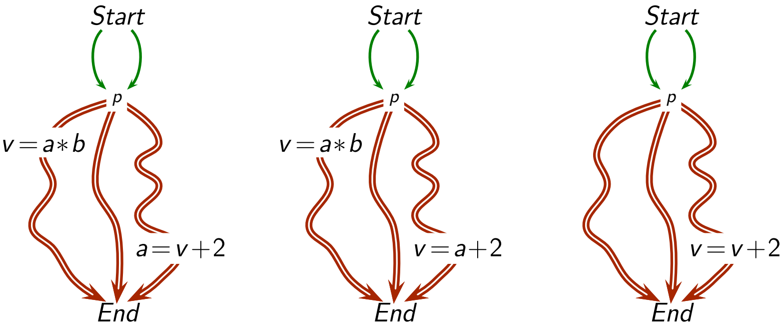
- Live Variables Analysis
- Some Observations
- Strongly Live Variables Analysis

Part 2

Live Variables Analysis

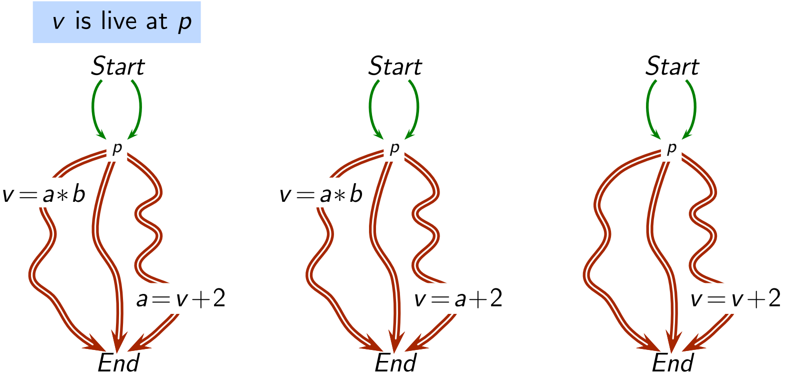
Defining Live Variables Analysis

A variable v is live at a program point p, if some path from p to program exit contains an r-value occurrence of v which is not preceded by an l-value occurrence of v.



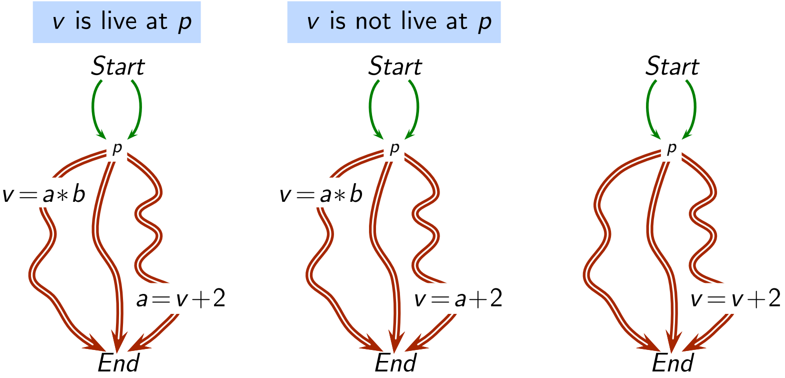
Defining Live Variables Analysis

A variable v is live at a program point p, if some path from p to program exit contains an r-value occurrence of v which is not preceded by an l-value occurrence of v.



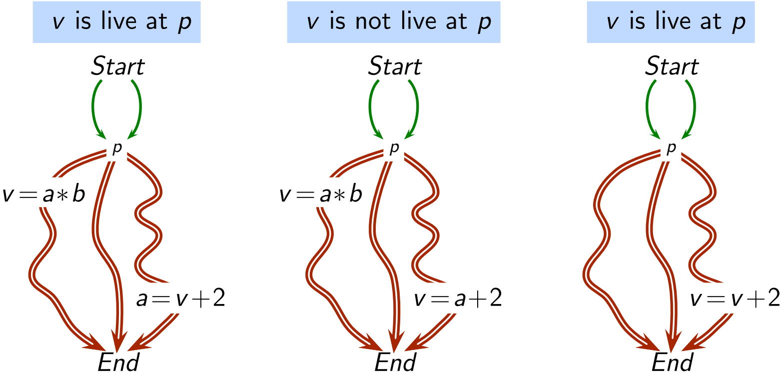
Defining Live Variables Analysis

A variable v is live at a program point p, if some path from p to program exit contains an r-value occurrence of v which is not preceded by an l-value occurrence of v.



Defining Live Variables Analysis

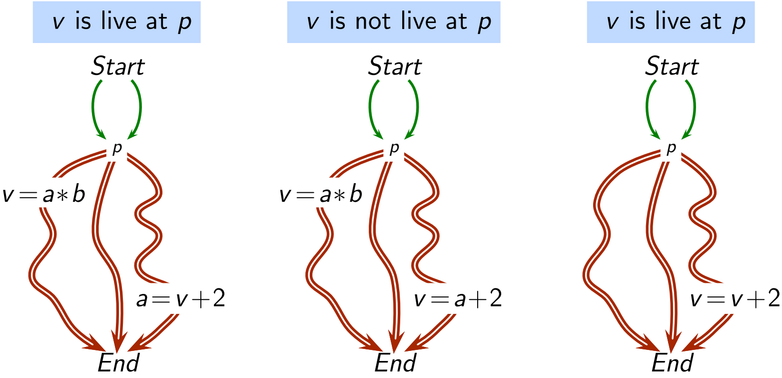
A variable v is live at a program point p, if some path from p to program exit contains an r-value occurrence of v which is not preceded by an l-value occurrence of v.



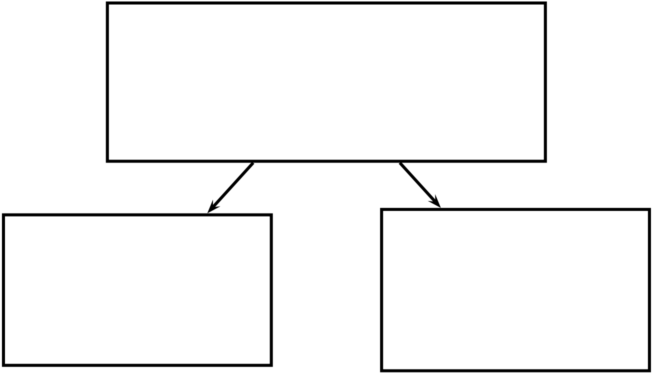
Defining Live Variables Analysis

A variable v is live at a program point p, if some path from p to program exit contains an r-value occurrence of v which is not preceded by an l-value occurrence of v.

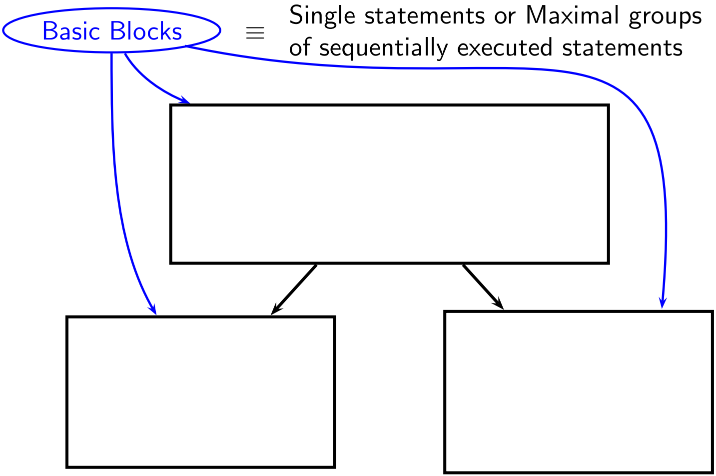
Path based specification



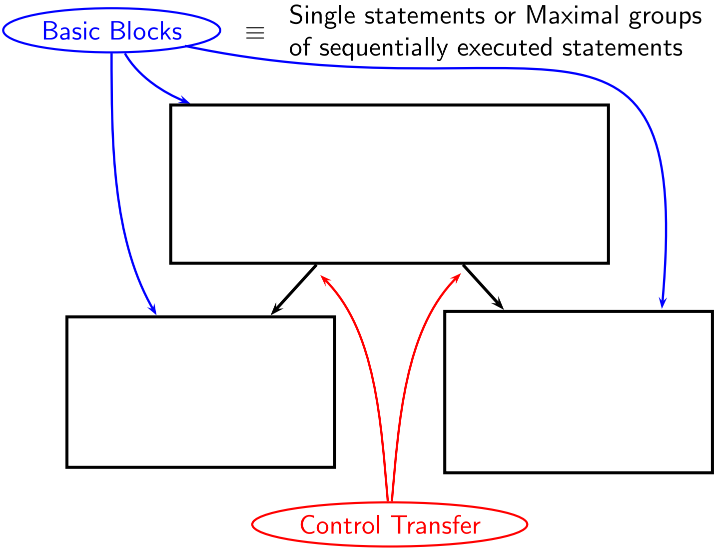
Defining Data Flow Analysis for Live Variables Analysis



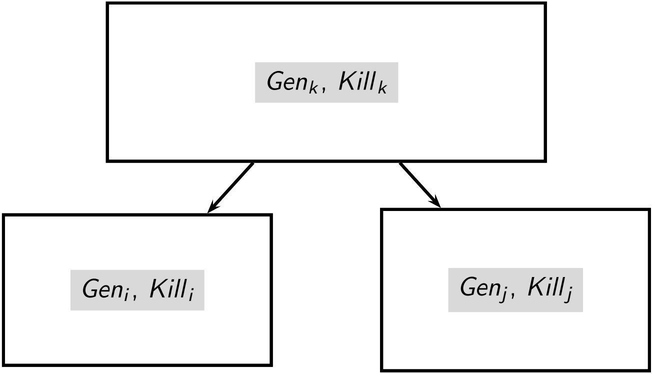
Defining Data Flow Analysis for Live Variables Analysis



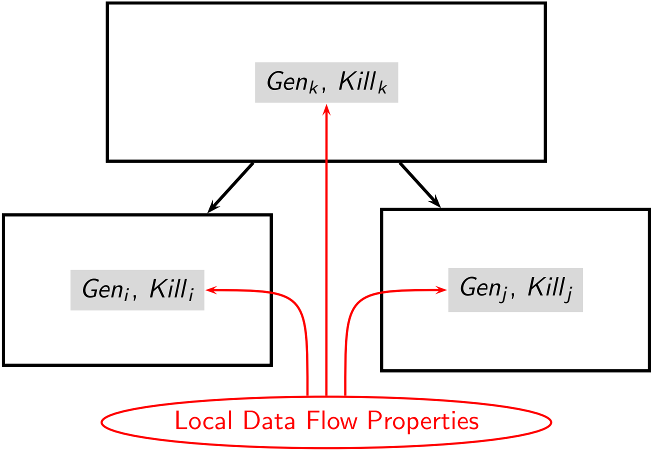
Defining Data Flow Analysis for Live Variables Analysis



Defining Data Flow Analysis for Live Variables Analysis



Defining Data Flow Analysis for Live Variables Analysis



Local Data Flow Properties for Live Variables Analysis

$$
\begin{aligned}
Gen_n &= \{\ v \mid \text{variable } v \text{ is used in basic block } n \text{ and} \\
&\qquad\quad \text{is not preceded by a definition of } v\ \} \\
Kill_n &= \{\ v \mid \text{basic block } n \text{ contains a definition of } v\ \}
\end{aligned}
$$

Local Data Flow Properties for Live Variables Analysis

r-value occurrence

Value is only read, e.g. x,y,z in

x.sum = y.data + z.data

$$Gen_n = \{ v \mid \text{variable } v \text{ is used in basic block } n \text{ and is not preceded by a definition of } v \}$$

$$Kill_n = \{ v \mid \text{basic block } n \text{ contains a definition of } v \}$$

Local Data Flow Properties for Live Variables Analysis

r-value occurrence

Value is only read, e.g. x,y,z in

x.sum = y.data + z.data

l-value occurrence

Value is modified e.g. y in

y = x.lptr

$$Gen_n = \{ v \mid \text{variable } v \text{ is used in basic block } n \text{ and is not preceded by a definition of } v \}$$

$$Kill_n = \{ v \mid \text{basic block } n \text{ contains a definition of } v \}$$

Local Data Flow Properties for Live Variables Analysis

r-value occurrence

Value is only read, e.g. x,y,z in

`x.sum = y.data + z.data`

l-value occurrence

Value is modified e.g. y in

`y = x.lptr`

$$Gen_n = \{ v \mid \text{variable } v \text{ is used in basic block } n \text{ and is not preceded by a definition of } v \}$$

$$Kill_n = \{ v \mid \text{basic block } n \text{ contains a definition of } v \}$$

within n

Local Data Flow Properties for Live Variables Analysis

r-value occurrence

Value is only read, e.g. x,y,z in

`x.sum = y.data + z.data`

l-value occurrence

Value is modified e.g. y in

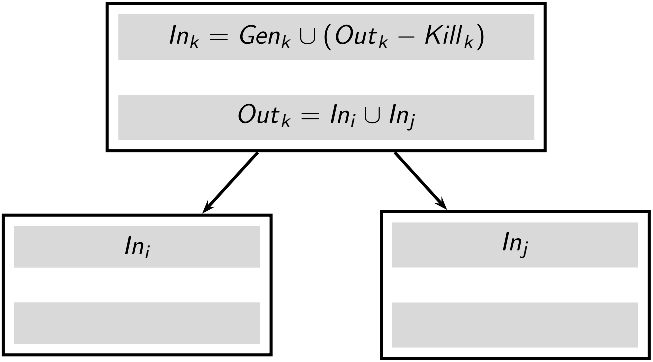
`y = x.lptr`

$$Gen_n = \{ v \mid \text{variable } v \text{ is used in basic block } n \text{ and is not preceded by a definition of } v \}$$

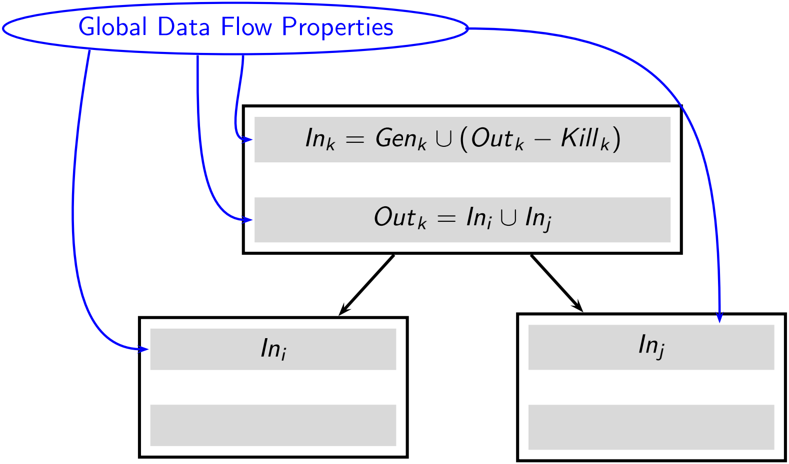
$$Kill_n = \{ v \mid \text{basic block } n \text{ contains a definition of } v \}$$

- "used": r-value occurrence
- "definition": l-value occurrence
- "preceded": within n
- "contains": anywhere in n

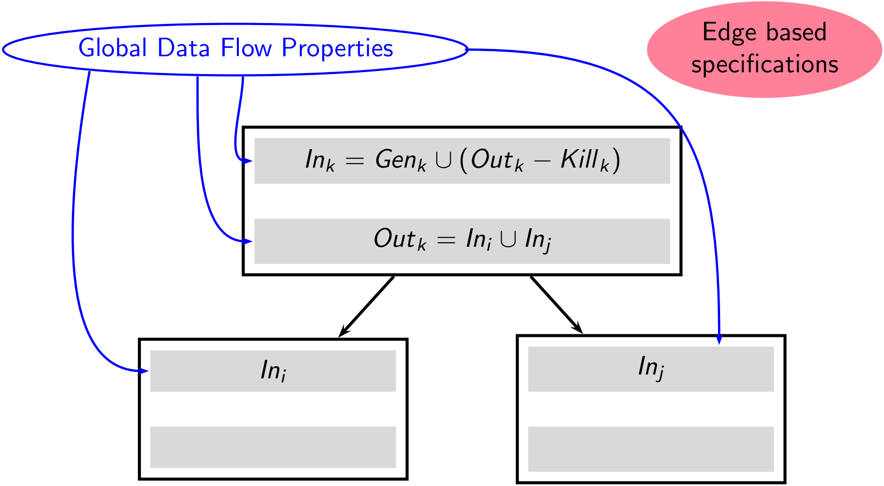
Defining Data Flow Analysis for Live Variables Analysis



Defining Data Flow Analysis for Live Variables Analysis



Defining Data Flow Analysis for Live Variables Analysis



Data Flow Equations For Live Variables Analysis

$$In_n = (Out_n - Kill_n) \cup Gen_n$$

$$Out_n = \begin{cases} BI & n \text{ is } End \text{ block} \\ \bigcup\limits_{s \in succ(n)} In_s & \text{otherwise} \end{cases}$$

Data Flow Equations For Live Variables Analysis

$$
\begin{aligned}
In_n &= (Out_n - Kill_n) \cup Gen_n \\
Out_n &= \begin{cases} BI & n \text{ is } End \text{ block} \\ \bigcup\limits_{s \in succ(n)} In_s & \text{otherwise} \end{cases}
\end{aligned}
$$

- In_n and Out_n are sets of variables

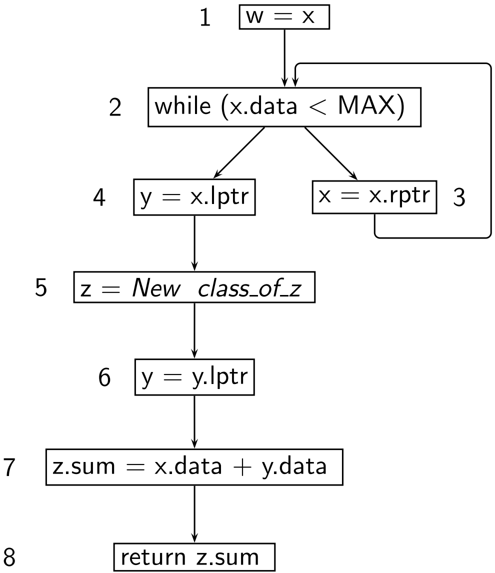
Data Flow Equations For Live Variables Analysis

$$In_n = (Out_n - Kill_n) \cup Gen_n$$

$$Out_n = \begin{cases} BI & n \text{ is } End \text{ block} \\ \bigcup_{s \in succ(n)} In_s & \text{otherwise} \end{cases}$$

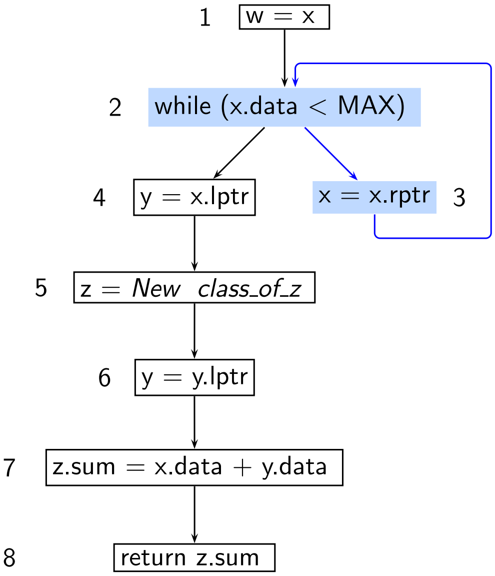
- In_n and Out_n are sets of variables
- BI is boundary information representing the effect of calling contexts
  - $\emptyset$ for local variables except for the values being returned
  - set of global variables used further in any calling context (can be safely approximated by the set of all global variables)

Data Flow Equations for Our Example



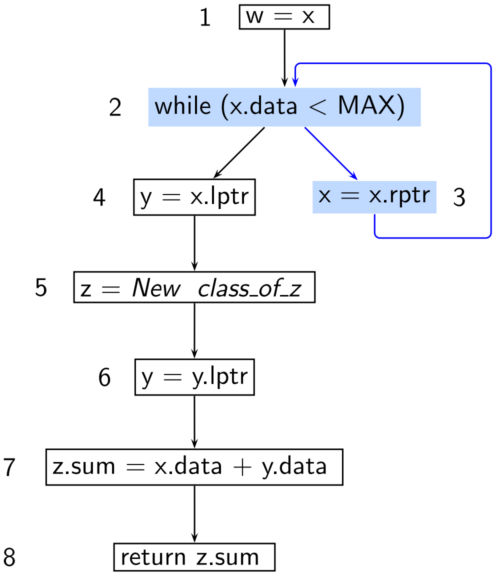
$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1 \\
Out_1 &= In_2 \\
In_2 &= (Out_2 - Kill_2) \cup Gen_2 \\
Out_2 &= In_3 \cup In_4 \\
In_3 &= (Out_3 - Kill_3) \cup Gen_3 \\
Out_3 &= In_2 \\
In_4 &= (Out_4 - Kill_4) \cup Gen_4 \\
Out_4 &= In_5 \\
In_5 &= (Out_5 - Kill_5) \cup Gen_5 \\
Out_5 &= In_6 \\
In_6 &= (Out_6 - Kill_6) \cup Gen_6 \\
Out_6 &= In_7 \\
In_7 &= (Out_7 - Kill_7) \cup Gen_7 \\
Out_7 &= In_8 \\
In_8 &= (Out_8 - Kill_8) \cup Gen_8 \\
Out_8 &= \emptyset
\end{aligned}
$$

Data Flow Equations for Our Example



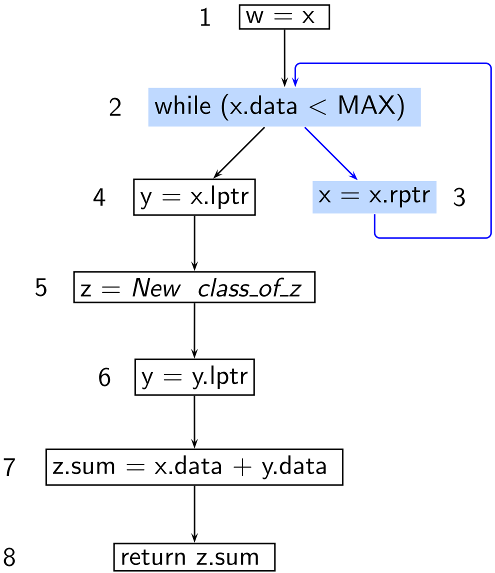
$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1 \\
Out_1 &= In_2 \\
In_2 &= (Out_2 - Kill_2) \cup Gen_2 \\
Out_2 &= In_3 \cup In_4 \\
In_3 &= (Out_3 - Kill_3) \cup Gen_3 \\
Out_3 &= In_2 \\
In_4 &= (Out_4 - Kill_4) \cup Gen_4 \\
Out_4 &= In_5 \\
In_5 &= (Out_5 - Kill_5) \cup Gen_5 \\
Out_5 &= In_6 \\
In_6 &= (Out_6 - Kill_6) \cup Gen_6 \\
Out_6 &= In_7 \\
In_7 &= (Out_7 - Kill_7) \cup Gen_7 \\
Out_7 &= In_8 \\
In_8 &= (Out_8 - Kill_8) \cup Gen_8 \\
Out_8 &= \emptyset
\end{aligned}
$$

Data Flow Equations for Our Example



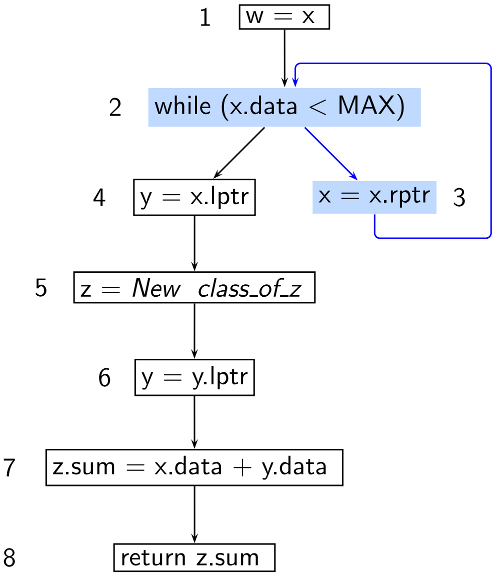
$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1\\
Out_1 &= In_2\\
In_2 &= (Out_2 - Kill_2) \cup Gen_2\\
Out_2 &= In_3 \cup In_4\\
In_3 &= (Out_3 - Kill_3) \cup Gen_3\\
Out_3 &= In_2\\
In_4 &= (Out_4 - Kill_4) \cup Gen_4\\
Out_4 &= In_5\\
In_5 &= (Out_5 - Kill_5) \cup Gen_5\\
Out_5 &= In_6\\
In_6 &= (Out_6 - Kill_6) \cup Gen_6\\
Out_6 &= In_7\\
In_7 &= (Out_7 - Kill_7) \cup Gen_7\\
Out_7 &= In_8\\
In_8 &= (Out_8 - Kill_8) \cup Gen_8\\
Out_8 &= \emptyset
\end{aligned}
$$

Data Flow Equations for Our Example



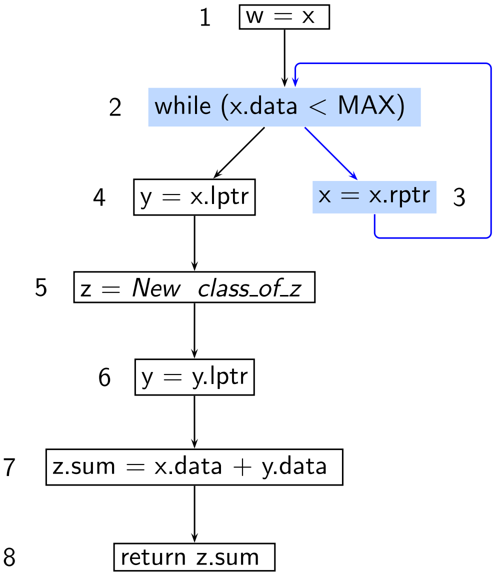
$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1 \\
Out_1 &= In_2 \\
In_2 &= (Out_2 - Kill_2) \cup Gen_2 \\
Out_2 &= In_3 \cup In_4 \\
In_3 &= (Out_3 - Kill_3) \cup Gen_3 \\
Out_3 &= In_2 \\
In_4 &= (Out_4 - Kill_4) \cup Gen_4 \\
Out_4 &= In_5 \\
In_5 &= (Out_5 - Kill_5) \cup Gen_5 \\
Out_5 &= In_6 \\
In_6 &= (Out_6 - Kill_6) \cup Gen_6 \\
Out_6 &= In_7 \\
In_7 &= (Out_7 - Kill_7) \cup Gen_7 \\
Out_7 &= In_8 \\
In_8 &= (Out_8 - Kill_8) \cup Gen_8 \\
Out_8 &= \emptyset
\end{aligned}
$$

Data Flow Equations for Our Example



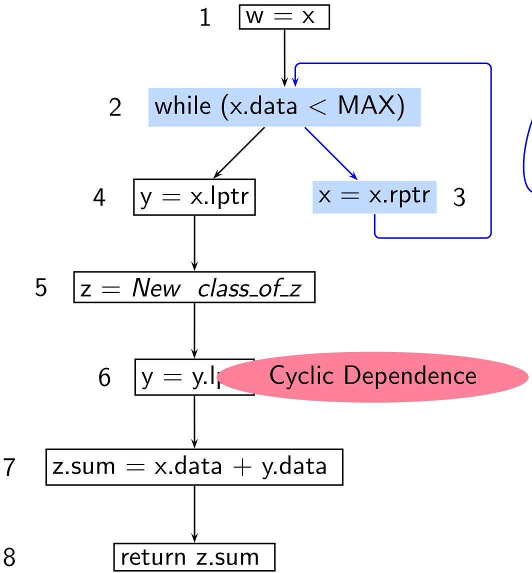
$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1 \\
Out_1 &= In_2 \\
In_2 &= (Out_2 - Kill_2) \cup Gen_2 \\
Out_2 &= In_3 \cup In_4 \\
In_3 &= (Out_3 - Kill_3) \cup Gen_3 \\
Out_3 &= In_2 \\
In_4 &= (Out_4 - Kill_4) \cup Gen_4 \\
Out_4 &= In_5 \\
In_5 &= (Out_5 - Kill_5) \cup Gen_5 \\
Out_5 &= In_6 \\
In_6 &= (Out_6 - Kill_6) \cup Gen_6 \\
Out_6 &= In_7 \\
In_7 &= (Out_7 - Kill_7) \cup Gen_7 \\
Out_7 &= In_8 \\
In_8 &= (Out_8 - Kill_8) \cup Gen_8 \\
Out_8 &= \emptyset
\end{aligned}
$$

Data Flow Equations for Our Example



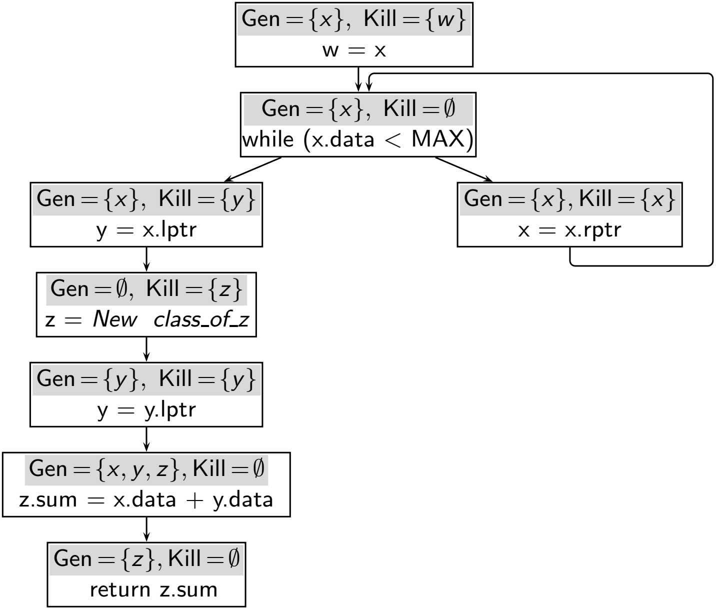
$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1 \\
Out_1 &= In_2 \\
In_2 &= (Out_2 - Kill_2) \cup Gen_2 \\
Out_2 &= In_3 \cup In_4 \\
In_3 &= (Out_3 - Kill_3) \cup Gen_3 \\
Out_3 &= In_2 \\
In_4 &= (Out_4 - Kill_4) \cup Gen_4 \\
Out_4 &= In_5 \\
In_5 &= (Out_5 - Kill_5) \cup Gen_5 \\
Out_5 &= In_6 \\
In_6 &= (Out_6 - Kill_6) \cup Gen_6 \\
Out_6 &= In_7 \\
In_7 &= (Out_7 - Kill_7) \cup Gen_7 \\
Out_7 &= In_8 \\
In_8 &= (Out_8 - Kill_8) \cup Gen_8 \\
Out_8 &= \emptyset
\end{aligned}
$$

Data Flow Equations for Our Example

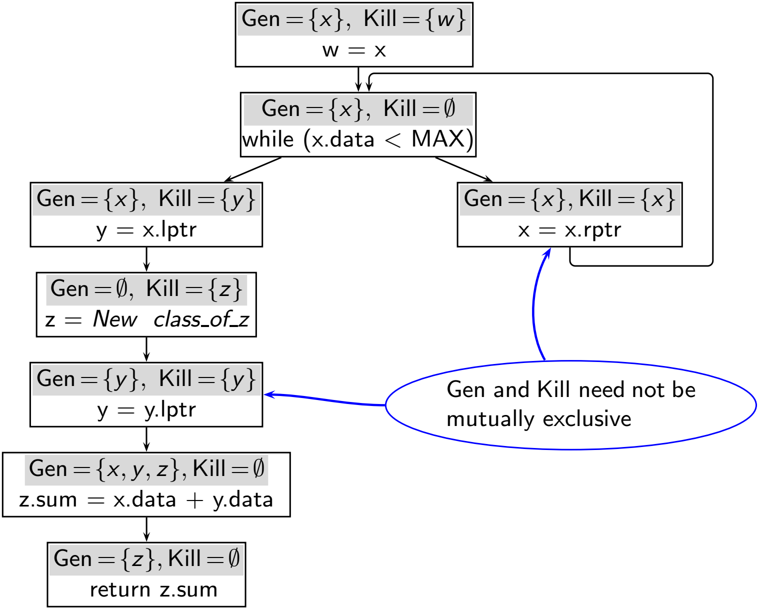


$$
\begin{aligned}
In_1 &= (Out_1 - Kill_1) \cup Gen_1 \\
Out_1 &= In_2 \\
In_2 &= (Out_2 - Kill_2) \cup Gen_2 \\
Out_2 &= In_3 \cup In_4 \\
In_3 &= (Out_3 - Kill_3) \cup Gen_3 \\
Out_3 &= In_2 \\
In_4 &= (Out_4 - Kill_4) \cup Gen_4 \\
Out_4 &= In_5 \\
In_5 &= (Out_5 - Kill_5) \cup Gen_5 \\
Out_5 &= In_6 \\
In_6 &= (Out_6 - Kill_6) \cup Gen_6 \\
Out_6 &= In_7 \\
In_7 &= (Out_7 - Kill_7) \cup Gen_7 \\
Out_7 &= In_8 \\
In_8 &= (Out_8 - Kill_8) \cup Gen_8 \\
Out_8 &= \emptyset
\end{aligned}
$$

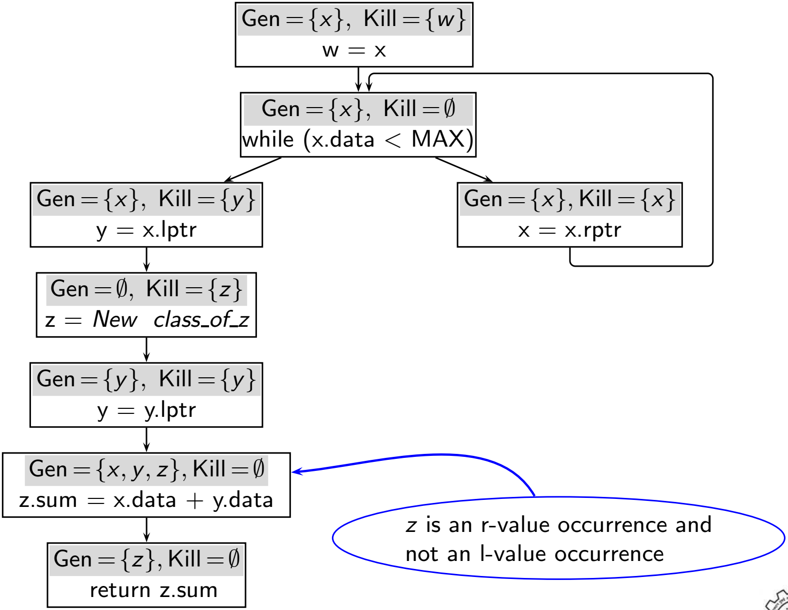
Performing Live Variables Analysis



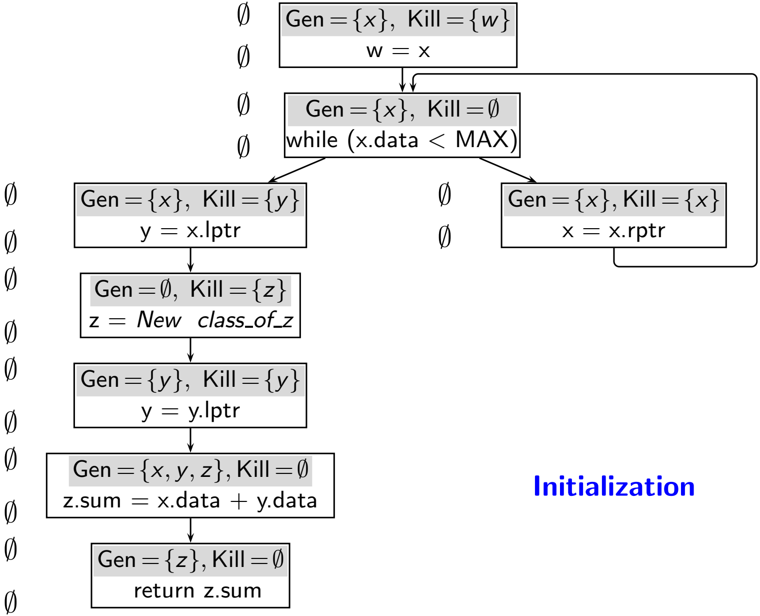
Performing Live Variables Analysis



Performing Live Variables Analysis

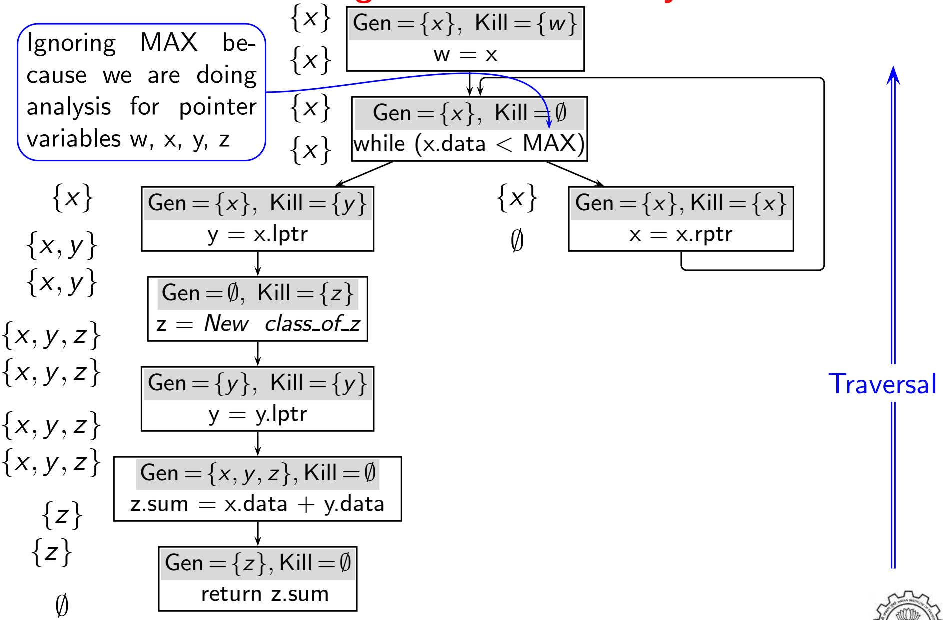


Performing Live Variables Analysis

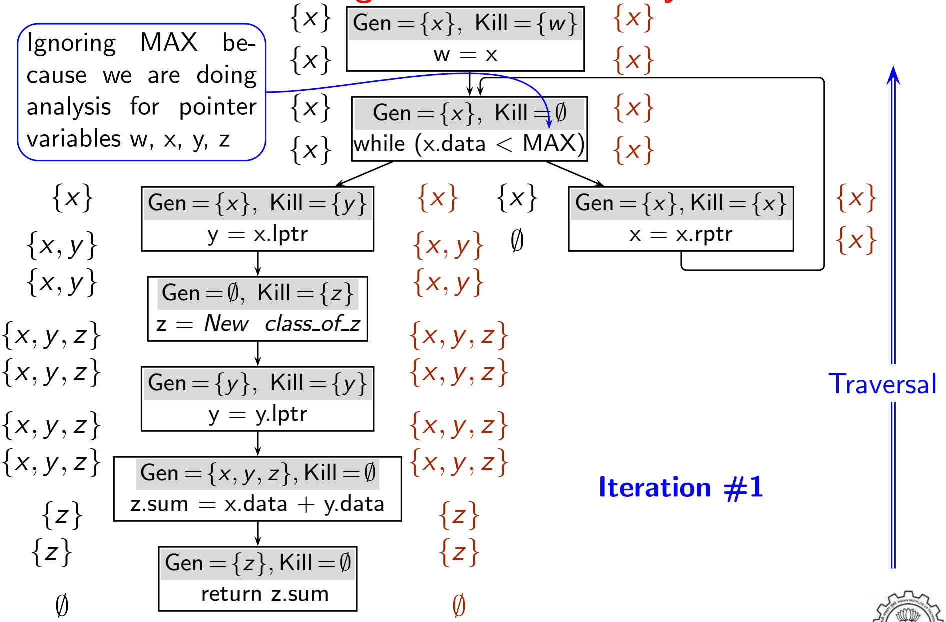


Initialization

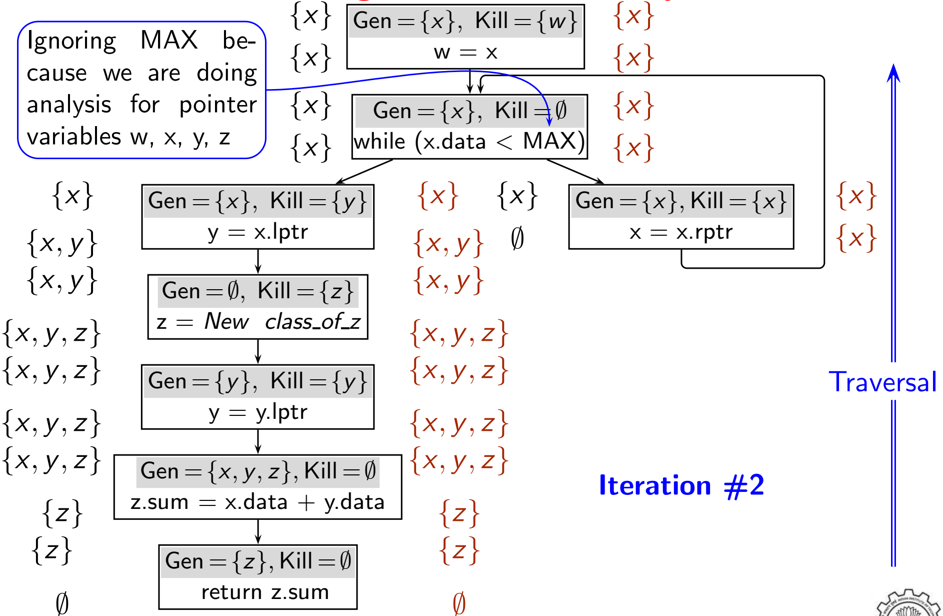
Performing Live Variables Analysis



Performing Live Variables Analysis

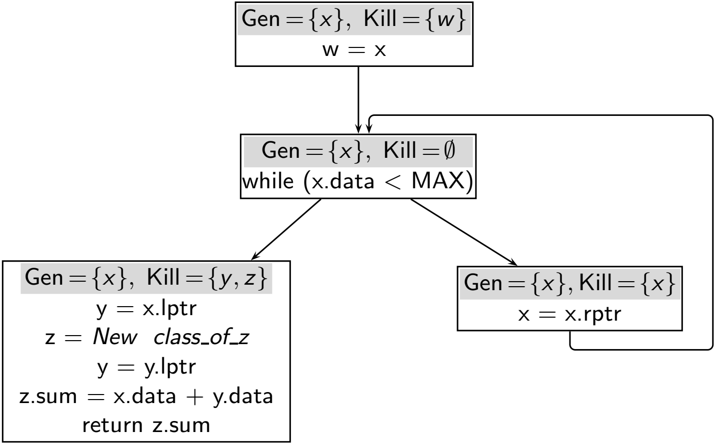


Performing Live Variables Analysis



Performing Live Variables Analysis

Local data flow properties when basic blocks contain multiple statements



Local Data Flow Properties for Live Variables Analysis

$$In_n = Gen_n \cup (Out_n - Kill_n)$$

- Gen_n : Use not preceded by definition
- $Kill_n$: Definition anywhere in a block

Local Data Flow Properties for Live Variables Analysis

$$In_n = Gen_n \cup (Out_n - Kill_n)$$

- Gen_n : Use not preceded by definition

  Upwards exposed use

- $Kill_n$: Definition anywhere in a block

  Stop the effect from being propagated across a block

Local Data Flow Properties for Live Variables Analysis

Case	Local Information		Example basic block	Explanation
1	$v \notin Gen_n$	$v \notin Kill_n$		
2	$v \in Gen_n$	$v \notin Kill_n$		
3	$v \notin Gen_n$	$v \in Kill_n$		
4	$v \in Gen_n$	$v \in Kill_n$		

Local Data Flow Properties for Live Variables Analysis

Case	Local Information		Example basic block	Explanation
1	$v \notin Gen_n$	$v \notin Kill_n$	$a = b + c$ $b = c * d$	liveness of v is unaffected by the basic block
2	$v \in Gen_n$	$v \notin Kill_n$	$a = b + c$ $b = v * d$	v becomes live before the basic block
3	$v \notin Gen_n$	$v \in Kill_n$	$a = b + c$ $v = c * d$ OR $v = a + b$ $c = v * d$	v ceases to be live before the basic block
4	$v \in Gen_n$	$v \in Kill_n$	$a = v + c$ $v = c * d$	liveness of v is killed but v becomes live before the basic block

Using Data Flow Information of Live Variables Analysis

- Used for register allocation

  If variable x is live in a basic block b, it is a potential candidate for register allocation

Using Data Flow Information of Live Variables Analysis

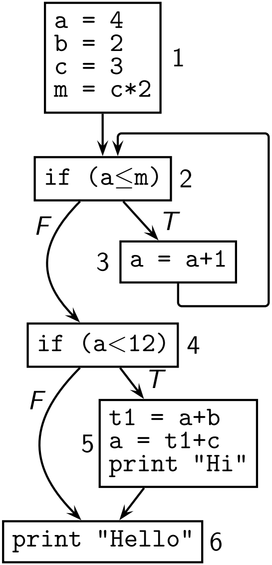
- Used for register allocation

  If variable x is live in a basic block b, it is a potential candidate for register allocation

- Used for dead code elimination

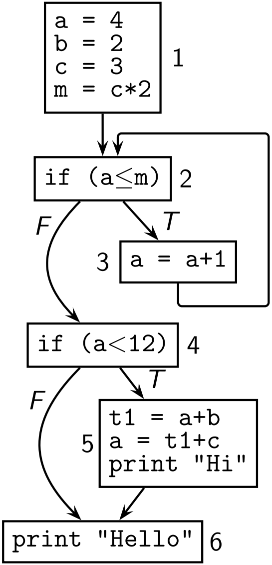
  If variable x is not live after an assignment $x = \ldots$, then the assignment is redundant and can be deleted as dead code

Tutorial Problem 1: Perform Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b, c\}$	$\{a, t1\}$
6	$\emptyset$	$\emptyset$

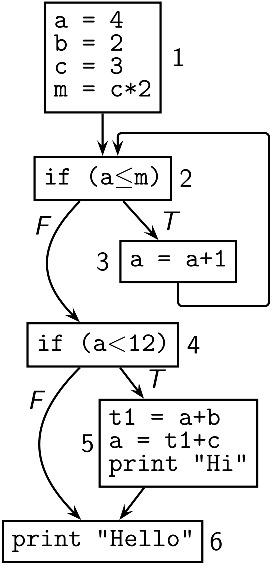
Tutorial Problem 1: Perform Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b, c\}$	$\{a, t1\}$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$		
5	$\emptyset$	$\{a, b, c\}$		
4	$\{a, b, c\}$	$\{a, b, c\}$		
3	$\emptyset$	$\{a\}$		
2	$\{a, b, c\}$	$\{a, b, c, m\}$		
1	$\{a, b, c, m\}$	$\emptyset$		

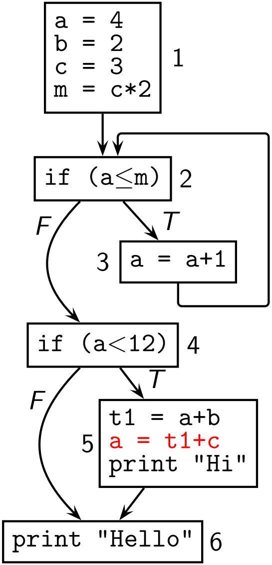
Tutorial Problem 1: Perform Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b, c\}$	$\{a, t1\}$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\{a, b, c\}$	$\emptyset$	$\{a, b, c\}$
4	$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, c\}$
3	$\emptyset$	$\{a\}$	$\{a, b, c, m\}$	$\{a, b, c, m\}$
2	$\{a, b, c\}$	$\{a, b, c, m\}$	$\{a, b, c, m\}$	$\{a, b, c, m\}$
1	$\{a, b, c, m\}$	$\emptyset$	$\{a, b, c, m\}$	$\emptyset$

Tutorial Problem 1: Perform Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b, c\}$	$\{a, t1\}$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\{a, b, c\}$	$\emptyset$	$\{a, b, c\}$
4	$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, c\}$
3	$\emptyset$	$\{a\}$	$\{a, b, c, m\}$	$\{a, b, c, m\}$
2	$\{a, b, c\}$	$\{a, b, c, m\}$	$\{a, b, c, m\}$	$\{a, b, c, m\}$
1	$\{a, b, c, m\}$	$\emptyset$	$\{a, b, c, m\}$	$\emptyset$

Tutorial Problem 1: Observeations About Round #1 of Dead Code Elimination

- We can repeat liveness analysis on the optimized code and then optimize it further

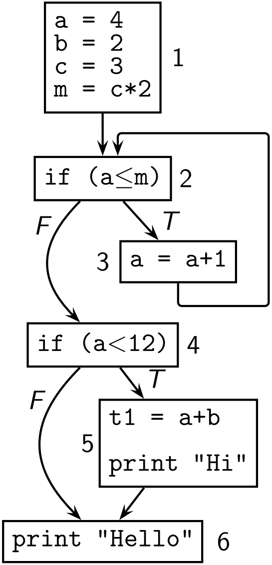
  This can continue as long code continues to change

- A better approach would be to perform strong liveness analysis

  The code needs to be optimized only once

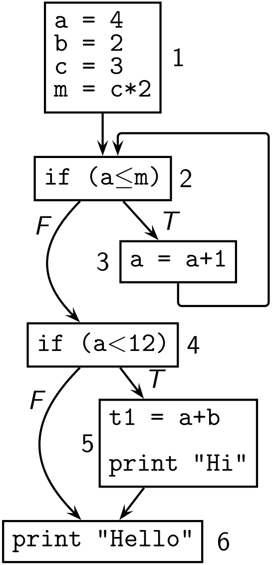
- Here we show the repeated application only to show the scope of further optimizations

Tutorial Problem 1: Round #2 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b\}$	$\{t1\}$
6	$\emptyset$	$\emptyset$

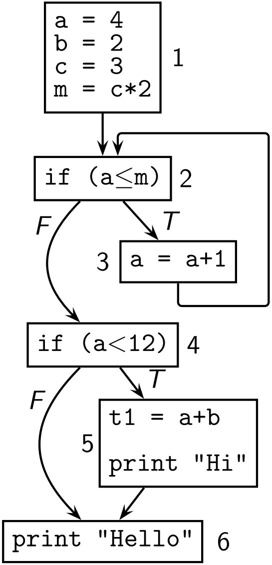
Tutorial Problem 1: Round #2 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b\}$	$\{t1\}$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$		
5	$\emptyset$	$\{a, b\}$		
4	$\{a, b\}$	$\{a, b\}$		
3	$\emptyset$	$\{a\}$		
2	$\{a, b\}$	$\{a, b, m\}$		
1	$\{a, b, m\}$	$\emptyset$		

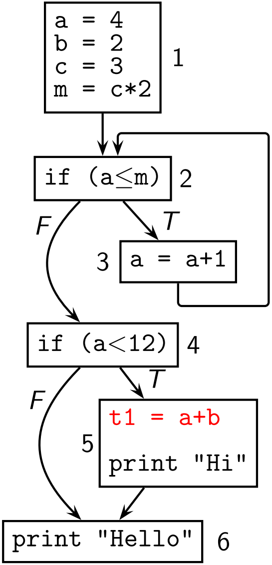
Tutorial Problem 1: Round #2 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b\}$	$\{t1\}$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\{a, b\}$	$\emptyset$	$\{a, b\}$
4	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$
3	$\emptyset$	$\{a\}$	$\{a, b, m\}$	$\{a, b, m\}$
2	$\{a, b\}$	$\{a, b, m\}$	$\{a, b, m\}$	$\{a, b, m\}$
1	$\{a, b, m\}$	$\emptyset$	$\{a, b, m\}$	$\emptyset$

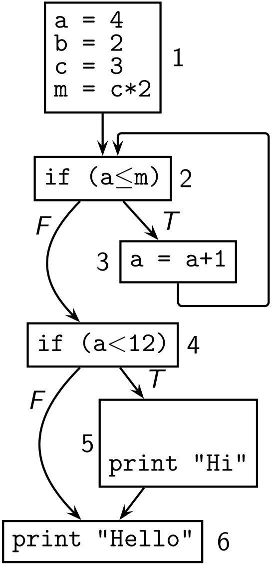
Tutorial Problem 1: Round #2 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\{a, b\}$	$\{t1\}$
6	$\emptyset$	$\emptyset$

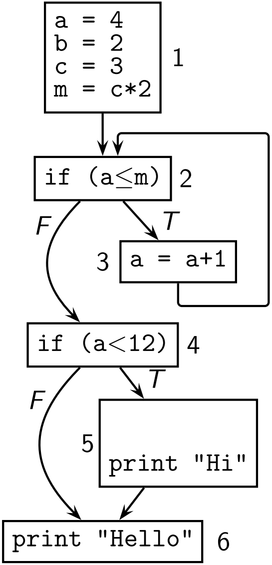
Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\{a, b\}$	$\emptyset$	$\{a, b\}$
4	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b\}$
3	$\emptyset$	$\{a\}$	$\{a, b, m\}$	$\{a, b, m\}$
2	$\{a, b\}$	$\{a, b, m\}$	$\{a, b, m\}$	$\{a, b, m\}$
1	$\{a, b, m\}$	$\emptyset$	$\{a, b, m\}$	$\emptyset$

Tutorial Problem 1: Round #3 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\emptyset$	$\emptyset$
6	$\emptyset$	$\emptyset$

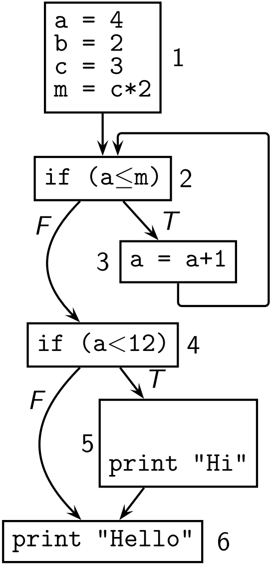
Tutorial Problem 1: Round #3 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\emptyset$	$\emptyset$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$		
5	$\emptyset$	$\emptyset$		
4	$\emptyset$	$\{a\}$		
3	$\emptyset$	$\{a\}$		
2	$\{a\}$	$\{a, m\}$		
1	$\{a, m\}$	$\emptyset$		

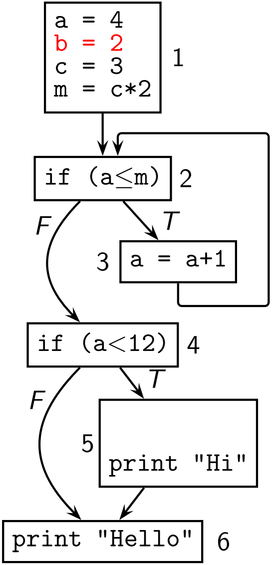
Tutorial Problem 1: Round #3 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\emptyset$	$\emptyset$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$\emptyset$	$\{a\}$	$\emptyset$	$\{a\}$
3	$\emptyset$	$\{a\}$	$\{a, m\}$	$\{a, m\}$
2	$\{a\}$	$\{a, m\}$	$\{a, m\}$	$\{a, m\}$
1	$\{a, m\}$	$\emptyset$	$\{a, m\}$	$\emptyset$

Tutorial Problem 1: Round #3 of Dead Code Elimination



Local Data Flow Information		
	Gen	*Kill*
1	$\emptyset$	$\{a, b, c, m\}$
2	$\{a, m\}$	$\emptyset$
3	$\{a\}$	$\{a\}$
4	$\{a\}$	$\emptyset$
5	$\emptyset$	$\emptyset$
6	$\emptyset$	$\emptyset$

Global Data Flow Information				
	Iteration #1		Iteration #2	
	Out	*In*	*Out*	*In*
6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$\emptyset$	$\{a\}$	$\emptyset$	$\{a\}$
3	$\emptyset$	$\{a\}$	$\{a, m\}$	$\{a, m\}$
2	$\{a\}$	$\{a, m\}$	$\{a, m\}$	$\{a, m\}$
1	$\{a, m\}$	$\emptyset$	$\{a, m\}$	$\emptyset$

Part 3

Some Observations

What Does Data Flow Analysis Involve?

- Defining the analysis.
- Formulating the analysis.
- Performing the analysis.

What Does Data Flow Analysis Involve?

- Defining the analysis. Define the properties of execution paths
- Formulating the analysis.
- Performing the analysis.

What Does Data Flow Analysis Involve?

- Defining the analysis. Define the properties of execution paths
- Formulating the analysis. Define data flow equations
  - Linear simultaneous equations on sets rather than numbers
  - Later we will generalize the domain of values
- Performing the analysis.

What Does Data Flow Analysis Involve?

- Defining the analysis. Define the properties of execution paths
- Formulating the analysis. Define data flow equations
  - Linear simultaneous equations on sets rather than numbers
  - Later we will generalize the domain of values
- Performing the analysis. Solve data flow equations for the given program flow graph

What Does Data Flow Analysis Involve?

- Defining the analysis. Define the properties of execution paths
- Formulating the analysis. Define data flow equations
  - Linear simultaneous equations on sets rather than numbers
  - Later we will generalize the domain of values
- Performing the analysis. Solve data flow equations for the given program flow graph
- Many unanswered questions

  Initial value? Termination? Complexity? Properties of Solutions?

Iterative Solution of Linear Simultaneous Equations

- Simultaneous equations represented in the form of the product of a matrix of coefficients (**A**) with the vector of unknowns (**x**)

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

- Start with approximate values
- Compute new values repeatedly from old values
- Two classical methods
  - Gauss-Seidel Method (Gauss: 1823, 1826), (Seidel: 1874)
  - Jacobi Method (Jacobi: 1845)

Our Method of Performing Data Flow Analysis

- Round robin iteration using the Jacobi method
  (use the values from the current iteration wherever possible)
- Unknowns are the data flow variables In_i and Out_i
- Domain of values is not numbers
- Computation in a fixed order
  - either forward (reverse post order) traversal, or
  - backward (post order) traversal

  over the control flow graph

Tutorial Problem 2 for Liveness Analysis

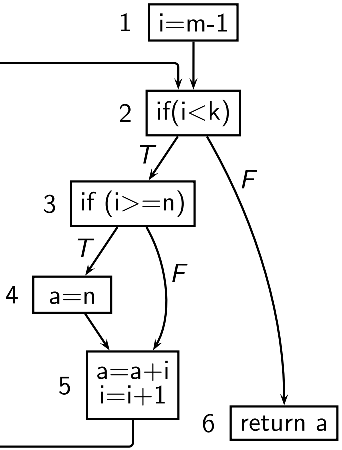
Draw the control flow graph and perform live variables analysis

```
int f(int m, int n, int k)
{
  int a,i;

  for (i=m-1; i<k; i++)
  {   if (i>=n)
          a = n;
      a = a+i;
  }
  return a;
}
```

Tutorial Problem 2 for Liveness Analysis

Draw the control flow graph and perform live variables analysis

```
int f(int m, int n, int k)
{
  int a,i;

  for (i=m-1; i<k; i++)
  {   if (i>=n)
          a = n;
      a = a+i;
  }
  return a;
}
```



Solution of Tutorial Problem 2

Block	Local Information		Global Information			
			Iteration # 1		Change in iteration # 2	
	Gen	*Kill*	*Out*	*In*	*Out*	*In*
6	$\{a\}$	$\emptyset$				
5	$\{a, i\}$	$\{a, i\}$				
4	$\{n\}$	$\{a\}$				
3	$\{i, n\}$	$\emptyset$				
2	$\{i, k\}$	$\emptyset$				
1	$\{m\}$	$\{i\}$				

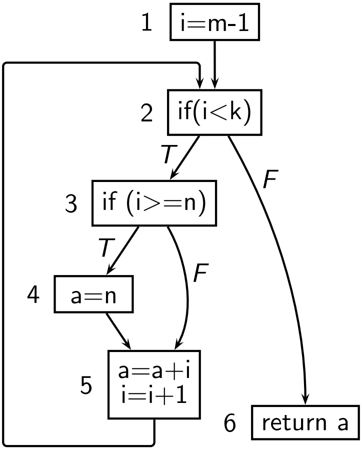
Solution of Tutorial Problem 2

Block	Local Information		Global Information			
			Iteration # 1		Change in iteration # 2	
	Gen	*Kill*	*Out*	*In*	*Out*	*In*
6	$\{a\}$	$\emptyset$	$\emptyset$	$\{a\}$		
5	$\{a, i\}$	$\{a, i\}$	$\emptyset$	$\{a, i\}$		
4	$\{n\}$	$\{a\}$	$\{a, i\}$	$\{i, n\}$		
3	$\{i, n\}$	$\emptyset$	$\{a, i, n\}$	$\{a, i, n\}$		
2	$\{i, k\}$	$\emptyset$	$\{a, i, n\}$	$\{a, i, k, n\}$		
1	$\{m\}$	$\{i\}$	$\{a, i, k, n\}$	$\{a, k, m, n\}$		

Solution of Tutorial Problem 2

Block	Local Information		Global Information			
			Iteration # 1		Change in iteration # 2	
	Gen	*Kill*	*Out*	*In*	*Out*	*In*
6	$\{a\}$	$\emptyset$	$\emptyset$	$\{a\}$		
5	$\{a, i\}$	$\{a, i\}$	$\emptyset$	$\{a, i\}$	$\{a, i, k, n\}$	$\{a, i, k, n\}$
4	$\{n\}$	$\{a\}$	$\{a, i\}$	$\{i, n\}$	$\{a, i, k, n\}$	$\{i, k, n\}$
3	$\{i, n\}$	$\emptyset$	$\{a, i, n\}$	$\{a, i, n\}$	$\{a, i, k, n\}$	$\{a, i, k, n\}$
2	$\{i, k\}$	$\emptyset$	$\{a, i, n\}$	$\{a, i, k, n\}$	$\{a, i, k, n\}$	
1	$\{m\}$	$\{i\}$	$\{a, i, k, n\}$	$\{a, k, m, n\}$		

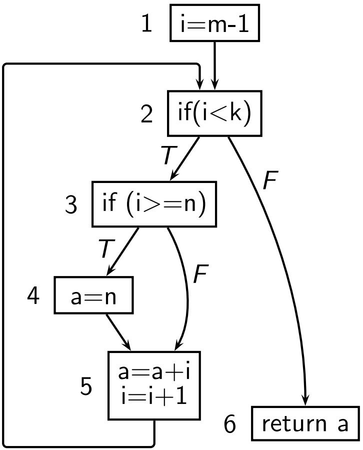
Interpreting the Result of Liveness Analysis for Tutorial Problem 2



- Is a live at the exit of node 5 at the end of iteration 1? Why?

  (We have used post order traversal)

Interpreting the Result of Liveness Analysis for Tutorial Problem 2



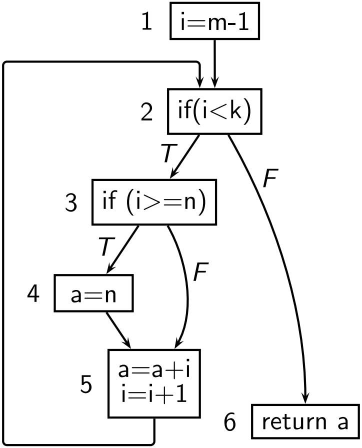
- Is a live at the exit of node 5 at the end of iteration 1? Why?

  (We have used post order traversal)

- Is a live at the exit of node 5 at the end of iteration 2? Why?

  (We have used post order traversal)

Interpreting the Result of Liveness Analysis for Tutorial Problem 2



- Is a live at the exit of node 5 at the end of iteration 1? Why?

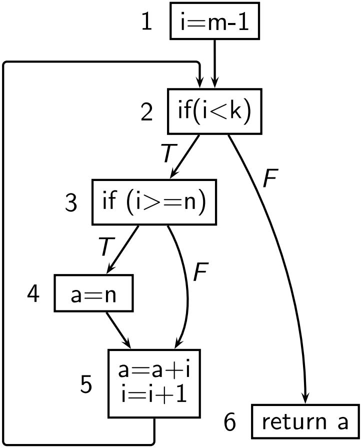
  (We have used post order traversal)

- Is a live at the exit of node 5 at the end of iteration 2? Why?

  (We have used post order traversal)

- Show an execution path along which a is live at the exit of node 5

Interpreting the Result of Liveness Analysis for Tutorial Problem 2



- Is a live at the exit of node 5 at the end of iteration 1? Why?

  (We have used post order traversal)

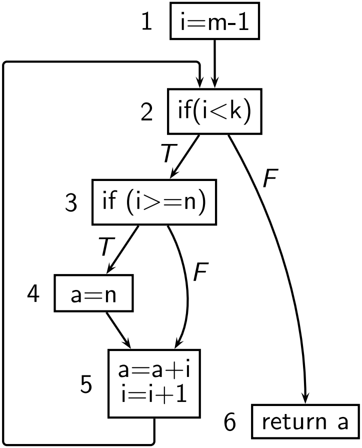
- Is a live at the exit of node 5 at the end of iteration 2? Why?

  (We have used post order traversal)

- Show an execution path along which a is live at the exit of node 5

- Show an execution path along which a is live at the exit of node 3

Interpreting the Result of Liveness Analysis for Tutorial Problem 2



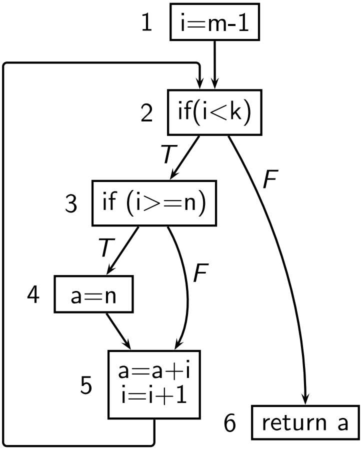
- Is a live at the exit of node 5 at the end of iteration 1? Why?

  (We have used post order traversal)
- Is a live at the exit of node 5 at the end of iteration 2? Why?

  (We have used post order traversal)
- Show an execution path along which a is live at the exit of node 5
- Show an execution path along which a is live at the exit of node 3

  $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow \dots$

Interpreting the Result of Liveness Analysis for Tutorial Problem 2



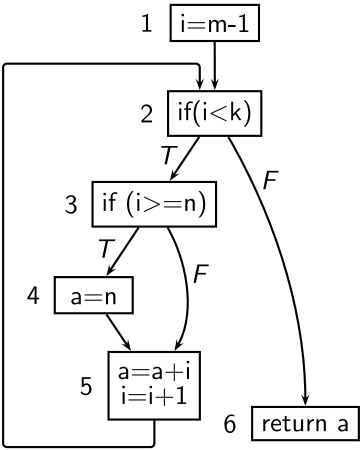
- Is a live at the exit of node 5 at the end of iteration 1? Why?

  (We have used post order traversal)
- Is a live at the exit of node 5 at the end of iteration 2? Why?

  (We have used post order traversal)
- Show an execution path along which a is live at the exit of node 5
- Show an execution path along which a is live at the exit of node 3

  $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow \ldots$
- Show an execution path along which a is not live at the exit of node 3

Interpreting the Result of Liveness Analysis for Tutorial Problem 2



- Is a live at the exit of node 5 at the end of iteration 1? Why?

  (We have used post order traversal)
- Is a live at the exit of node 5 at the end of iteration 2? Why?

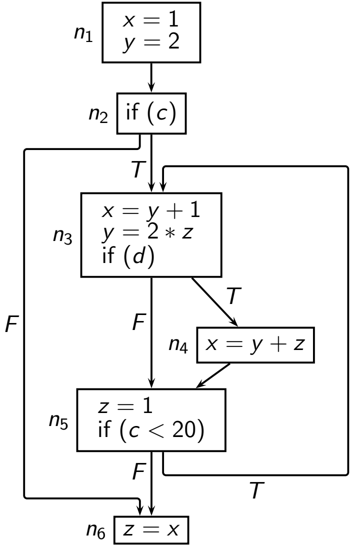
  (We have used post order traversal)
- Show an execution path along which a is live at the exit of node 5
- Show an execution path along which a is live at the exit of node 3

  $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow \ldots$
- Show an execution path along which a is not live at the exit of node 3

  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow \ldots$

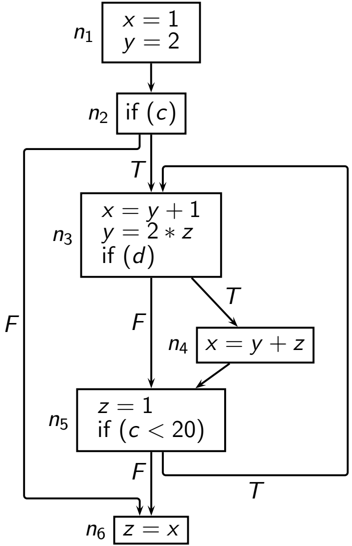
Tutorial Problem 3 for Liveness Analysis

Also write a C program for this CFG without using goto or break



Tutorial Problem 3 for Liveness Analysis

Also write a C program for this CFG without using goto or break




```
void f()
{  int x, y, z;
   int c, d;
   x = 1;
   y = 2;
   if (c)
   {   do
       {   x = y+1;
           y = 2*z;
           if (d)
               x = y+z;
           z = 1;
       } while (c < 20);
   }
   z = x;
}
```

Solution of Tutorial Problem 3

Block	Local Information		Global Information			
			Iteration # 1		Change in iteration # 2	
	Gen	*Kill*	*Out*	*In*	*Out*	*In*
n_6	$\{x\}$	$\{z\}$				
n_5	$\{c\}$	$\{z\}$				
n_4	$\{y, z\}$	$\{x\}$				
n_3	$\{y, z, d\}$	$\{x, y\}$				
n_2	$\{c\}$	$\emptyset$				
n_1	$\emptyset$	$\{x, y\}$				

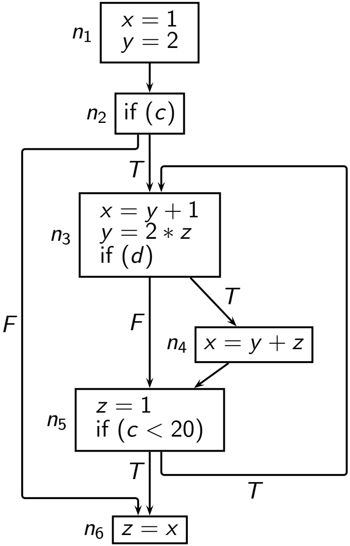
Solution of Tutorial Problem 3

Block	Local Information		Global Information			
			Iteration # 1		Change in iteration # 2	
	Gen	*Kill*	*Out*	*In*	*Out*	*In*
n_6	$\{x\}$	$\{z\}$	$\emptyset$	$\{x\}$		
n_5	$\{c\}$	$\{z\}$	$\{x\}$	$\{x, c\}$		
n_4	$\{y, z\}$	$\{x\}$	$\{x, c\}$	$\{y, z, c\}$		
n_3	$\{y, z, d\}$	$\{x, y\}$	$\{x, y, z, c\}$	$\{y, z, c, d\}$		
n_2	$\{c\}$	$\emptyset$	$\{x, y, z, c, d\}$	$\{x, y, z, c, d\}$		
n_1	$\emptyset$	$\{x, y\}$	$\{x, y, z, c, d\}$	$\{z, c, d\}$		

Solution of Tutorial Problem 3

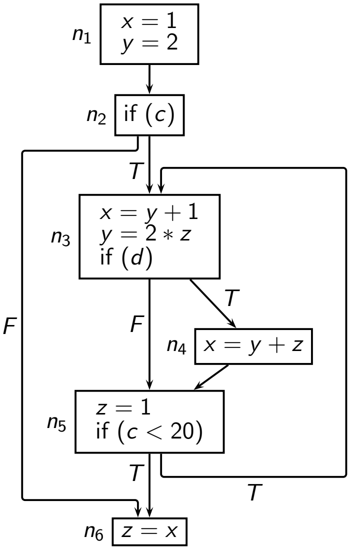
Block	Local Information		Global Information			
			Iteration # 1		Change in iteration # 2	
	Gen	*Kill*	*Out*	*In*	*Out*	*In*
n_6	$\{x\}$	$\{z\}$	$\emptyset$	$\{x\}$		
n_5	$\{c\}$	$\{z\}$	$\{x\}$	$\{x, c\}$	$\{x, y, z, c, d\}$	$\{x, y, c, d\}$
n_4	$\{y, z\}$	$\{x\}$	$\{x, c\}$	$\{y, z, c\}$	$\{x, y, c, d\}$	$\{y, z, c, d\}$
n_3	$\{y, z, d\}$	$\{x, y\}$	$\{x, y, z, c\}$	$\{y, z, c, d\}$	$\{x, y, z, c, d\}$	
n_2	$\{c\}$	$\emptyset$	$\{x, y, z, c, d\}$	$\{x, y, z, c, d\}$		
n_1	$\emptyset$	$\{x, y\}$	$\{x, y, z, c, d\}$	$\{z, c, d\}$		

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



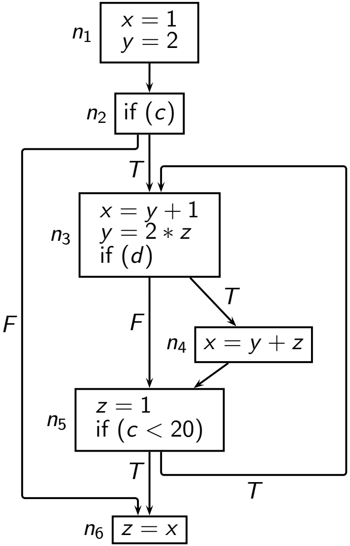
- Why is z live at the exit of n_5?

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



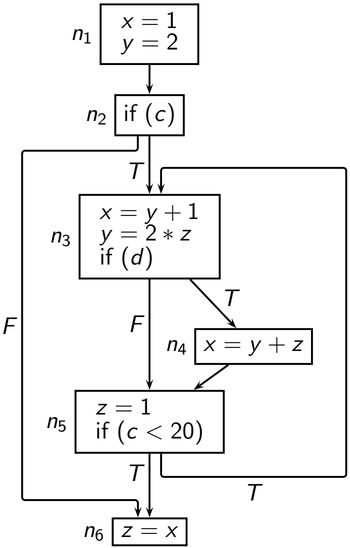
- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



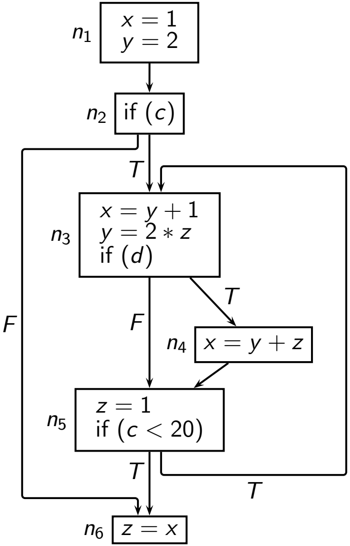
- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



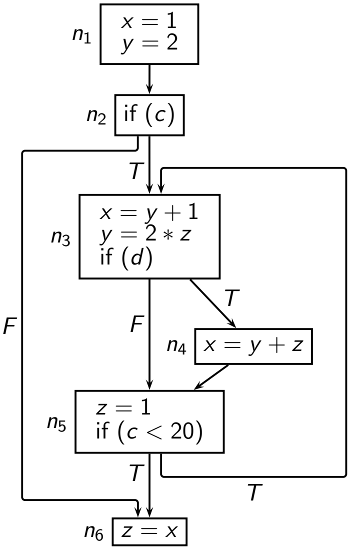
- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?
- Identify the instance of dead code elimination

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



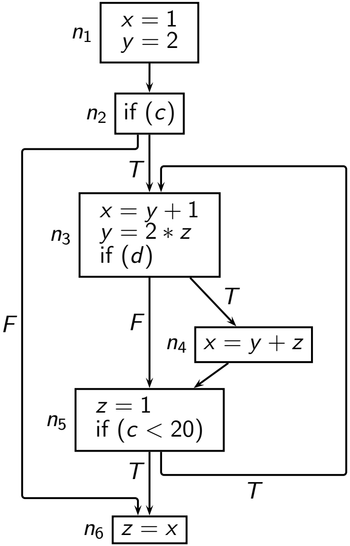
- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?
- Identify the instance of dead code elimination $z = x$ in n_6

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



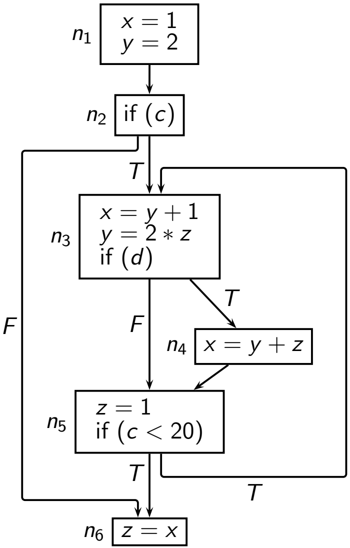
- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?
- Identify the instance of dead code elimination $z = x$ in n_6
- Would the first round of dead code elimination cause liveness information to change?

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



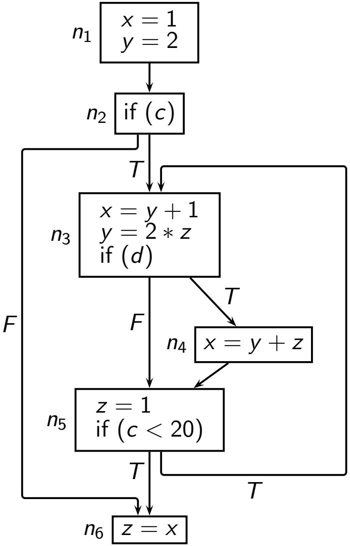
- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?
- Identify the instance of dead code elimination $z = x$ in n_6
- Would the first round of dead code elimination cause liveness information to change? Yes

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?
- Identify the instance of dead code elimination $z = x$ in n_6
- Would the first round of dead code elimination cause liveness information to change? Yes
- Would the second round of liveness analysis lead to further dead code elimination?

Interpreting the Result of Liveness Analysis for Tutorial Problem 3



- Why is z live at the exit of n_5?
- Why is z not live at the entry of n_5?
- Why is x live at the exit of n_3 inspite of being killed in n_4?
- Identify the instance of dead code elimination $z = x$ in n_6
- Would the first round of dead code elimination cause liveness information to change? Yes
- Would the second round of liveness analysis lead to further dead code elimination? Yes

Choice of Initialization

What should be the initial value of internal nodes?

Choice of Initialization

What should be the initial value of internal nodes?

- Confluence is $\cup$
- Identity of $\cup$ is $\emptyset$

Choice of Initialization

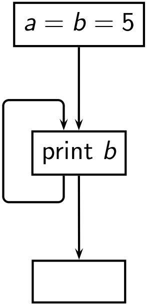
What should be the initial value of internal nodes?

- Confluence is $\cup$
- Identity of $\cup$ is $\emptyset$
- We begin with $\emptyset$ and let the sets at each program point grow

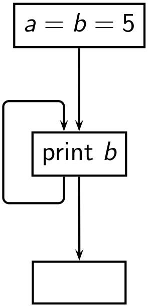
  A revisit to a program point

  - may consider a new execution path
  - more variables may be found to be live
  - a variable found to be live earlier does not become dead

How Does the Initialization Affect the Solution?



How Does the Initialization Affect the Solution?



Init.

$\emptyset$

$\emptyset$

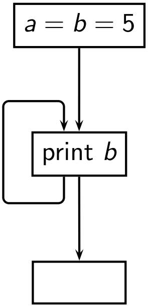
$\emptyset$

$\emptyset$

$\emptyset$

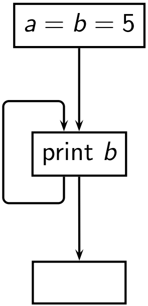
$\emptyset$

How Does the Initialization Affect the Solution?



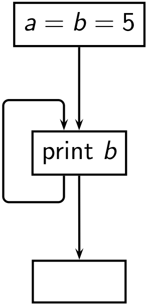
	Init.	Iter. #1
	$\emptyset$	
	$\emptyset$	
	$\emptyset$	
	$\emptyset$	
	$\emptyset$	
	$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?



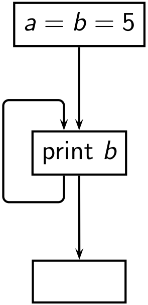
Init.	Iter. #1
$\emptyset$	
$\emptyset$	
$\emptyset$	
$\emptyset$	
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?



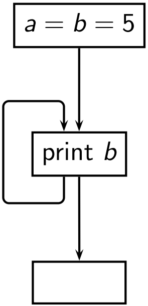
	Init.	Iter. #1
	$\emptyset$	
	$\emptyset$	
	$\emptyset$	
	$\emptyset$	$\emptyset$
	$\emptyset$	$\emptyset$
	$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?



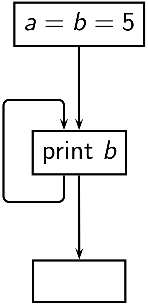
Init.	Iter. #1
$\emptyset$	
$\emptyset$	
$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?



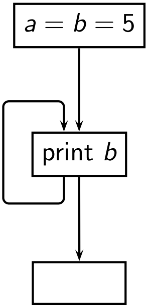
Init.	Iter. #1
$\emptyset$	
$\emptyset$	$\{b\}$
$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?



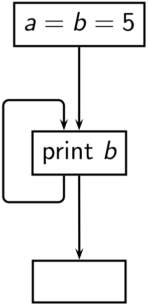
Init.	Iter. #1
$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$
$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

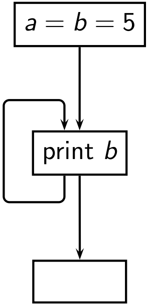


	Init.	Iter. #1	Iter. #2
	$\emptyset$	$\emptyset$	$\emptyset$
	$\emptyset$	$\{b\}$	$\{b\}$
	$\emptyset$	$\{b\}$	$\{b\}$
	$\emptyset$	$\emptyset$	$\{b\}$
	$\emptyset$	$\emptyset$	$\emptyset$
	$\emptyset$	$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

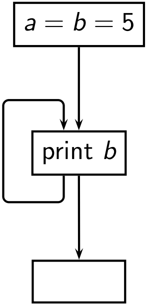


	Init.	Iter. #1	Iter. #2
	$\emptyset$	$\emptyset$	$\emptyset$
	$\emptyset$	$\{b\}$	$\{b\}$
	$\emptyset$	$\{b\}$	$\{b\}$
	$\emptyset$	$\emptyset$	$\{b\}$
	$\emptyset$	$\emptyset$	$\emptyset$
	$\emptyset$	$\emptyset$	$\emptyset$

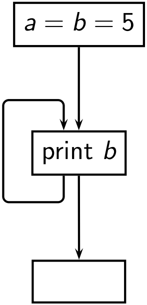


	Init.
	$\{a, b\}$
	$\{a, b\}$
	$\{a, b\}$
	$\{a, b\}$
	$\{a, b\}$
	$\emptyset$

How Does the Initialization Affect the Solution?

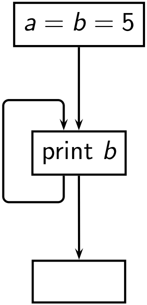


Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

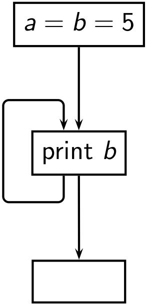


Init.	Iter. #1
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

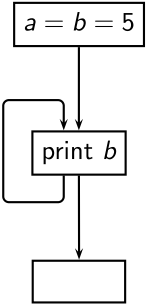


Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

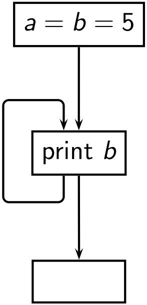


Init.	Iter. #1
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

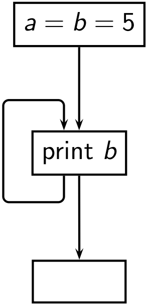


Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

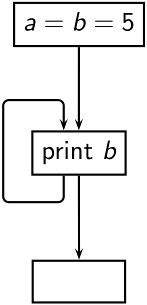


Init.	Iter. #1
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

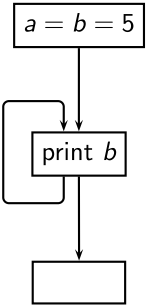


Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

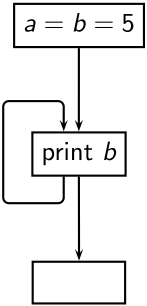


Init.	Iter. #1
$\{a, b\}$	
$\{a, b\}$	
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

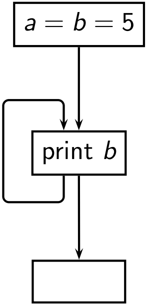


Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

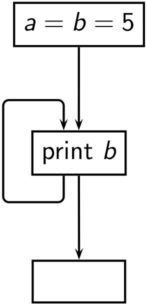


Init.	Iter. #1
$\{a, b\}$	
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?

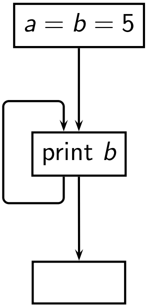


Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

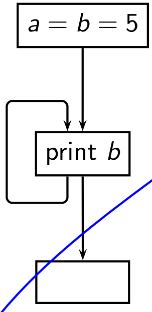


Init.	Iter. #1
$\{a, b\}$	$\emptyset$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\emptyset$
$\emptyset$	$\emptyset$

How Does the Initialization Affect the Solution?



Init.	Iter. #1	Iter. #2
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\{b\}$	$\{b\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$



Init.	Iter. #1
$\{a, b\}$	$\emptyset$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\{a, b\}$
$\{a, b\}$	$\emptyset$
$\emptyset$	$\emptyset$

a is spuriously marked live

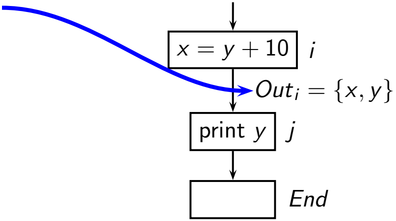
Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

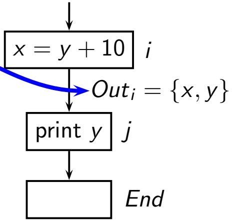
- Spurious inclusion of a non-live variable



Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

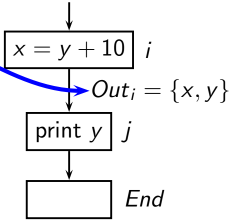
- Spurious inclusion of a non-live variable
  - A dead assignment may not be eliminated
  - Solution is sound but may be imprecise

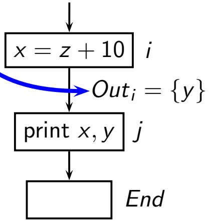


Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

- Spurious inclusion of a non-live variable
  - A dead assignment may not be eliminated
  - Solution is sound but may be imprecise
- Spurious exclusion of a live variable

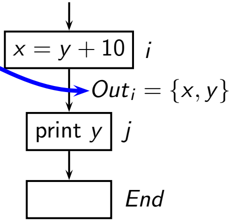


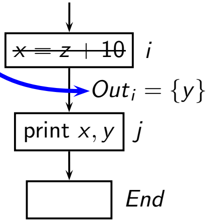


Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

- Spurious inclusion of a non-live variable
  - A dead assignment may not be eliminated
  - Solution is sound but may be imprecise
- Spurious exclusion of a live variable
  - A useful assignment may be eliminated
  - Solution is unsound

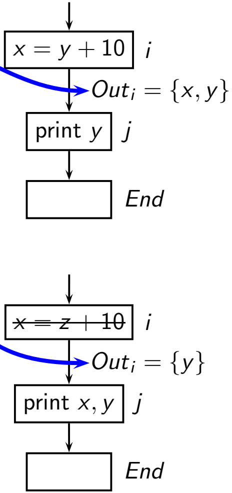




Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

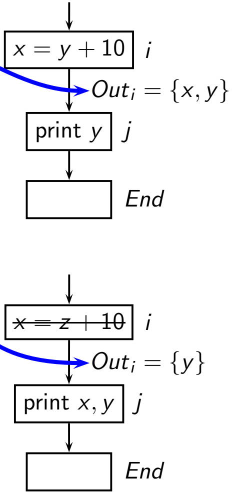
- Spurious inclusion of a non-live variable
  - A dead assignment may not be eliminated
  - Solution is sound but may be imprecise
- Spurious exclusion of a live variable
  - A useful assignment may be eliminated
  - Solution is unsound
- Given $L_2 \supseteq L_1$ representing liveness information
  - Using L_2 in place of L_1 is sound
  - Using L_1 in place of L_2 may not be sound



Soundness and Precision of Live Variables Analysis

Consider dead code elimination based on liveness information

- Spurious inclusion of a non-live variable
  - A dead assignment may not be eliminated
  - Solution is sound but may be imprecise
- Spurious exclusion of a live variable
  - A useful assignment may be eliminated
  - Solution is unsound
- Given $L_2 \supseteq L_1$ representing liveness information
  - Using L_2 in place of L_1 is sound
  - Using L_1 in place of L_2 may not be sound
- The smallest set of all live variables is most precise
  - Since liveness sets grow (confluence is $\cup$), we choose $\emptyset$ as the initial conservative value



Termination, Convergence, and Complexity

- For live variables analysis,
    - The set of all variables is finite, and
    - the confluence operation (i.e. meet) is union, hence
    - the set associated with a data flow variable can only grow

$\Longrightarrow$ Termination is guaranteed

Termination, Convergence, and Complexity

- For live variables analysis,
  - The set of all variables is finite, and
  - the confluence operation (i.e. meet) is union, hence
  - the set associated with a data flow variable can only grow

  $\Longrightarrow$ Termination is guaranteed

- Since initial value is $\emptyset$, live variables analysis converges on the smallest set

Termination, Convergence, and Complexity

- For live variables analysis,
  - The set of all variables is finite, and
  - the confluence operation (i.e. meet) is union, hence
  - the set associated with a data flow variable can only grow

  $\Longrightarrow$ Termination is guaranteed
- Since initial value is $\emptyset$, live variables analysis converges on the smallest set
- How many iterations do we need for reaching the convergence?

Termination, Convergence, and Complexity

- For live variables analysis,
  - The set of all variables is finite, and
  - the confluence operation (i.e. meet) is union, hence
  - the set associated with a data flow variable can only grow

  $\Rightarrow$ Termination is guaranteed
- Since initial value is $\emptyset$, live variables analysis converges on the smallest set
- How many iterations do we need for reaching the convergence?
- Going beyond live variables analysis
  - Do the sets always grow for other data flow frameworks?
  - What is the complexity of round robin analysis for other analyses?

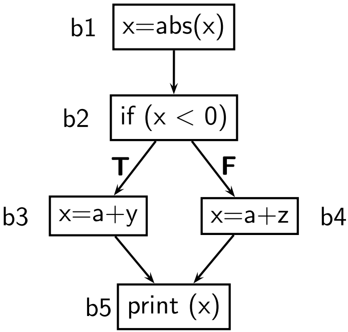
Termination, Convergence, and Complexity

- For live variables analysis,
  - The set of all variables is finite, and
  - the confluence operation (i.e. meet) is union, hence
  - the set associated with a data flow variable can only grow

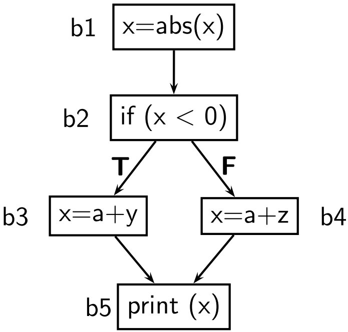
  $\Longrightarrow$ Termination is guaranteed
- Since initial value is $\emptyset$, live variables analysis converges on the smallest set
- How many iterations do we need for reaching the convergence?
- Going beyond live variables analysis
  - Do the sets always grow for other data flow frameworks?
  - What is the complexity of round robin analysis for other analyses?

  Answered formally in module 2 (Theoretical Abstractions)

Conservative Nature of Analysis (1)

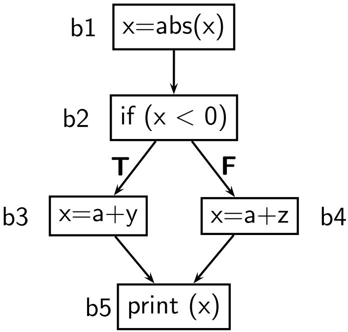


Conservative Nature of Analysis (1)



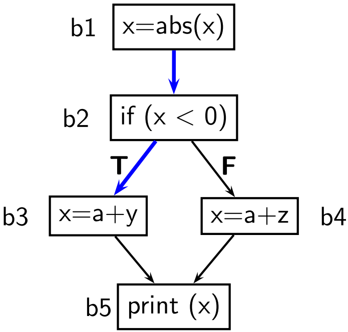
- abs(n) returns the absolute value of n

Conservative Nature of Analysis (1)



- abs(n) returns the absolute value of n
- Is y live on entry to block b2?

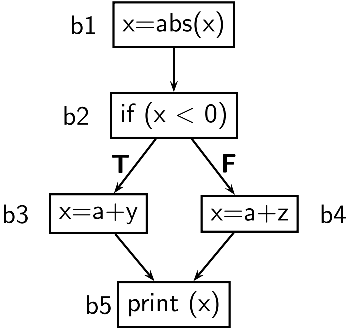
Conservative Nature of Analysis (1)



- abs(n) returns the absolute value of n
- Is y live on entry to block b2?
- By execution semantics, NO

  Path b1→b2→b3 is an infeasible execution path

Conservative Nature of Analysis (1)



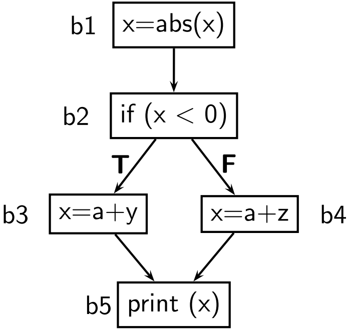
- abs(n) returns the absolute value of n
- Is y live on entry to block b2?
- By execution semantics, NO

  Path b1→b2→b3 is an infeasible execution path
- A compiler makes conservative assumptions:

  All branch outcomes are possible

  ⇒ Consider every path in CFG as a potential execution path

Conservative Nature of Analysis (1)



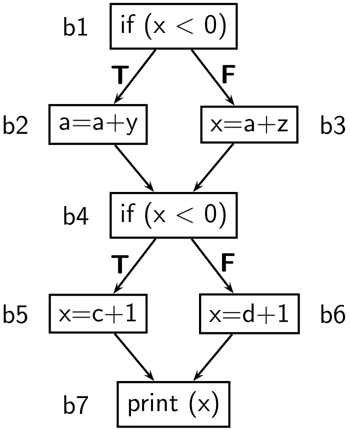
- abs(n) returns the absolute value of n
- Is y live on entry to block b2?
- By execution semantics, NO

  Path b1→b2→b3 is an infeasible execution path
- A compiler makes conservative assumptions:

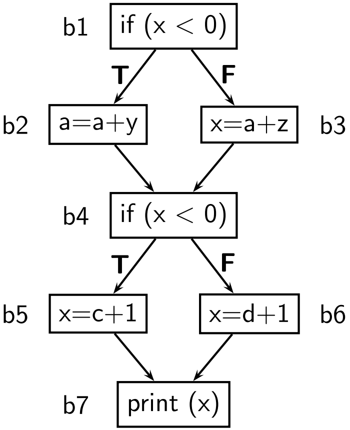
  All branch outcomes are possible

  ⇒ Consider every path in CFG as a potential execution path
- Our analysis concludes that y is live on entry to block b2

Conservative Nature of Analysis (2)

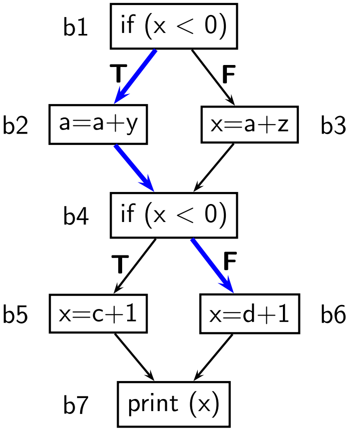


Conservative Nature of Analysis (2)



- Is d live on entry to block b2?

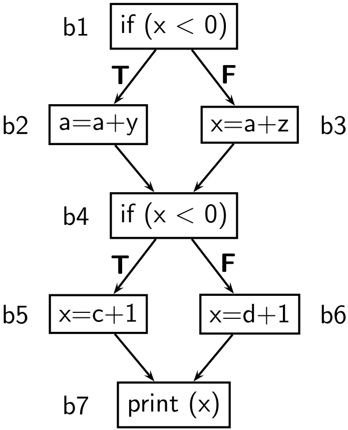
Conservative Nature of Analysis (2)



- Is d live on entry to block b2?
- By execution semantics, NO

  Path b1→b2→b4→b6 is an infeasible execution path

Conservative Nature of Analysis (2)

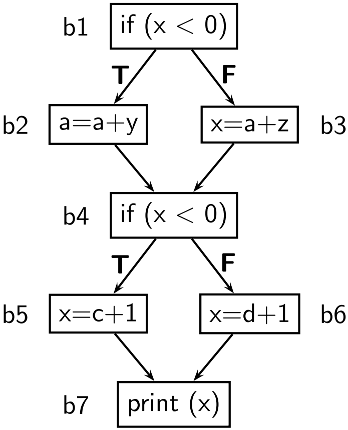


- Is d live on entry to block b2?
- By execution semantics, NO

  Path b1→b2→b4→b6 is an infeasible execution path
- Is c live on entry to block b3?

  Path b1→b3→b4→b6 is a feasible execution path

Conservative Nature of Analysis (2)



- Is d live on entry to block b2?
- By execution semantics, NO

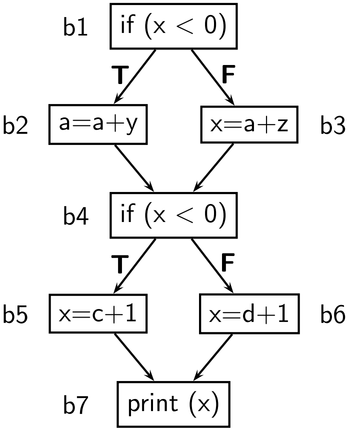
  Path b1→b2→b4→b6 is an infeasible execution path
- Is c live on entry to block b3?

  Path b1→b3→b4→b6 is a feasible execution path
- A compiler make conservative assumptions

  ⇒ our analysis is *path insensitive*

  Note: It is *flow sensitive* (i.e. information is computed for every control flow points)

Conservative Nature of Analysis (2)



- Is d live on entry to block b2?
- By execution semantics, NO

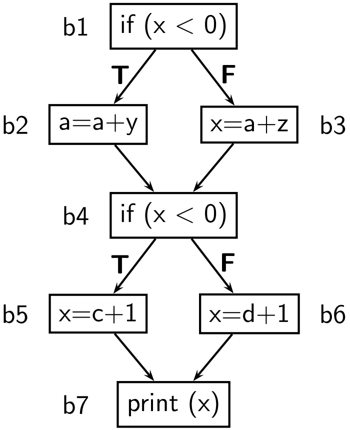
  Path b1→b2→b4→b6 is an infeasible execution path
- Is c live on entry to block b3?

  Path b1→b3→b4→b6 is a feasible execution path
- A compiler make conservative assumptions

  ⇒ our analysis is *path insensitive*

  Note: It is *flow sensitive* (i.e. information is computed for every control flow points)
- Our analysis concludes that d is live at the entry of b2

Conservative Nature of Analysis (2)



- Is d live on entry to block b2?
- By execution semantics, NO

  Path b1→b2→b4→b6 is an infeasible execution path
- Is c live on entry to block b3?

  Path b1→b3→b4→b6 is a feasible execution path
- A compiler make conservative assumptions $\Rightarrow$ our analysis is *path insensitive*

  Note: It is *flow sensitive* (i.e. information is computed for every control flow points)
- Our analysis concludes that d is live at the entry of b2
- Is c live at the entry of b3?

Conservative Nature of Analysis at Intraprocedural Level

- We assume that all paths are potentially executable
- Our analysis is path insensitive
    - The data flow information at a program point p is path insensitive
        - information at p is merged along all paths reaching p
    - The data flow information reaching p is computed path insensitively
        - information is merged at all shared points in paths reaching p
        - may generate spurious information due to non-distributive flow functions

Conservative Nature of Analysis at Interprocedural Level

- Context insensitivity
  - Merges of information across all calling contexts
- Flow insensitivity
  - Disregards the control flow

More about it later

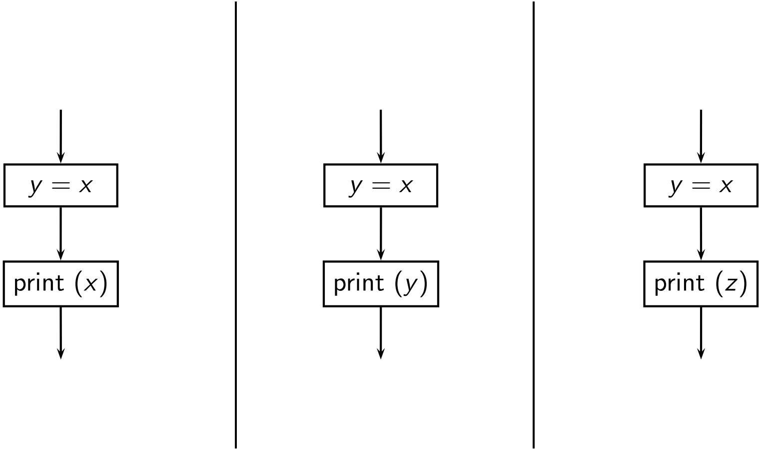
Part 4

Strongly Live Variables Analysis

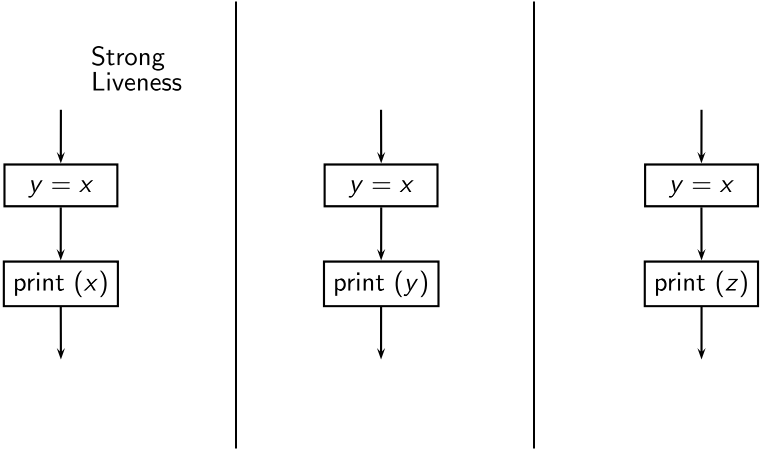
Strongly Live Variables Analysis

- A variable is strongly live if
  - it is used in a statement other than assignment statement, or
    (same as simple liveness)
  - it is used in an assignment statement defining a variable that is strongly live
    (different from simple liveness)
- Killing: An assignment statement, an input statement, or BI
  (this is same as killing in simple liveness)
- Generation: A direct use or a use for defining values that are strongly live
  (this is different from generation in simple liveness)

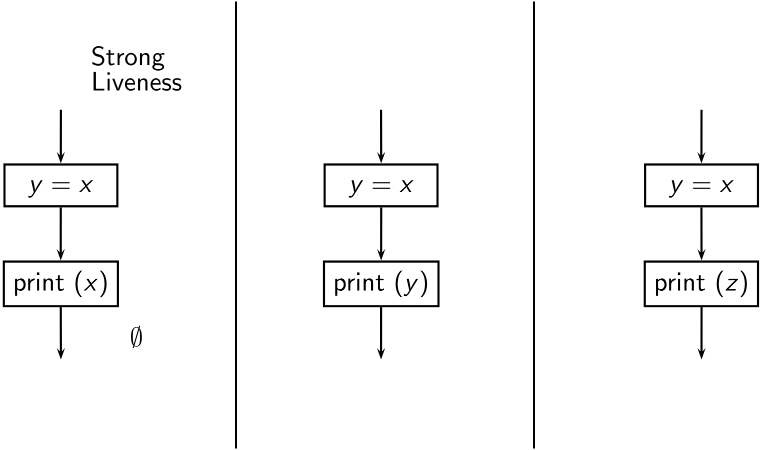
Understanding Strong Liveness



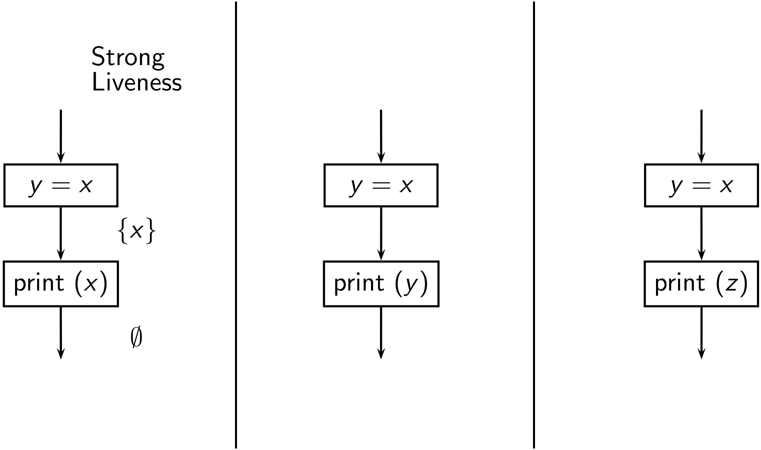
Understanding Strong Liveness



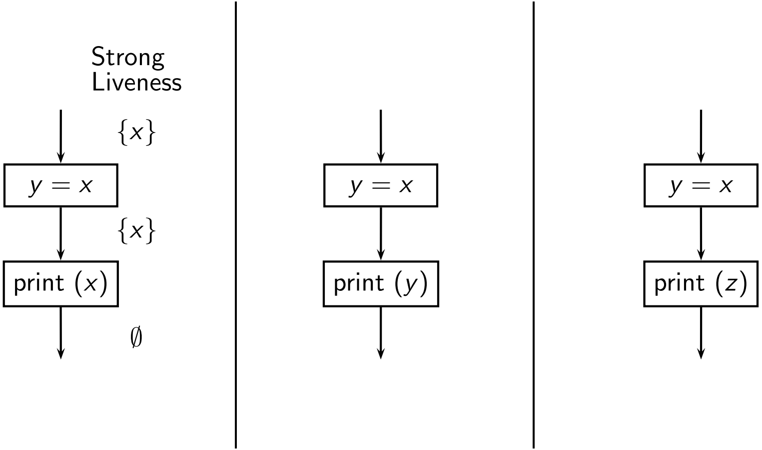
Understanding Strong Liveness



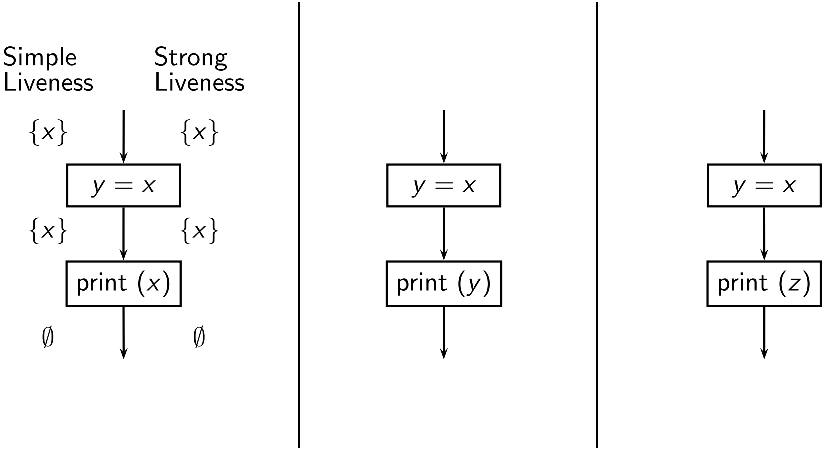
Understanding Strong Liveness



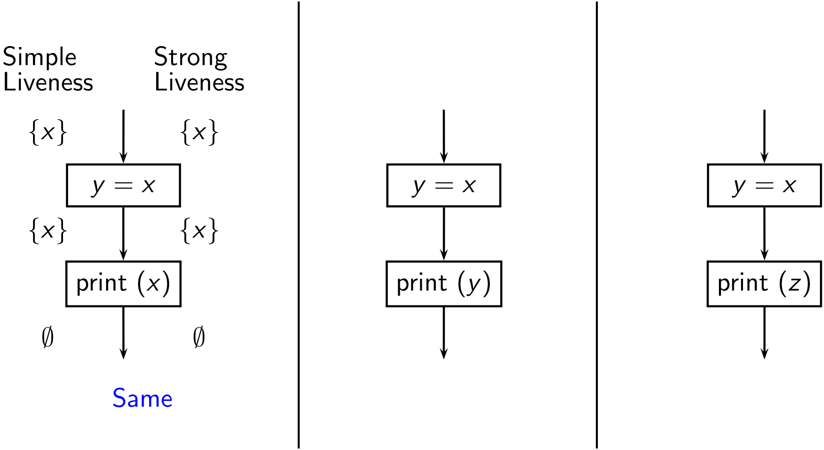
Understanding Strong Liveness



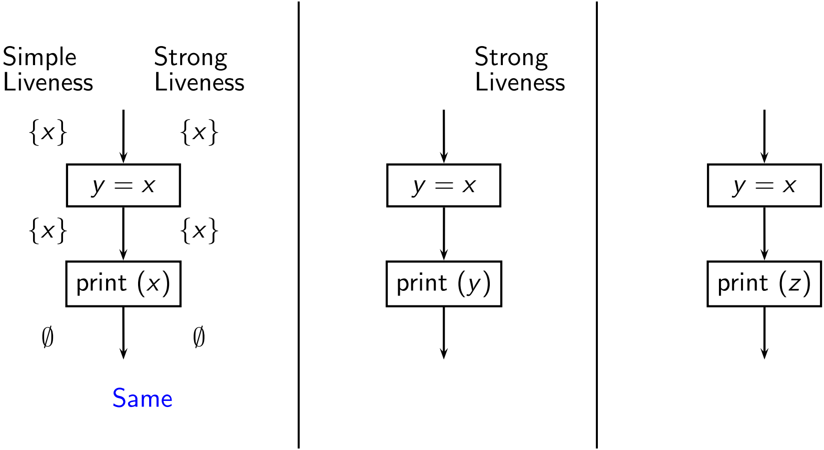
Understanding Strong Liveness



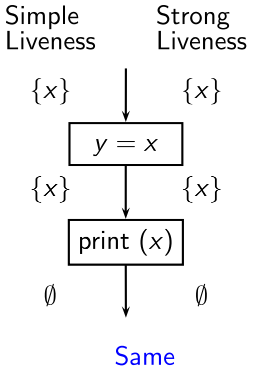
Understanding Strong Liveness

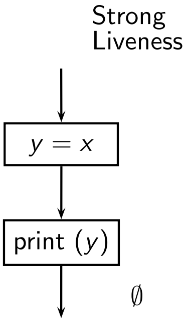


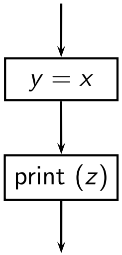
Understanding Strong Liveness



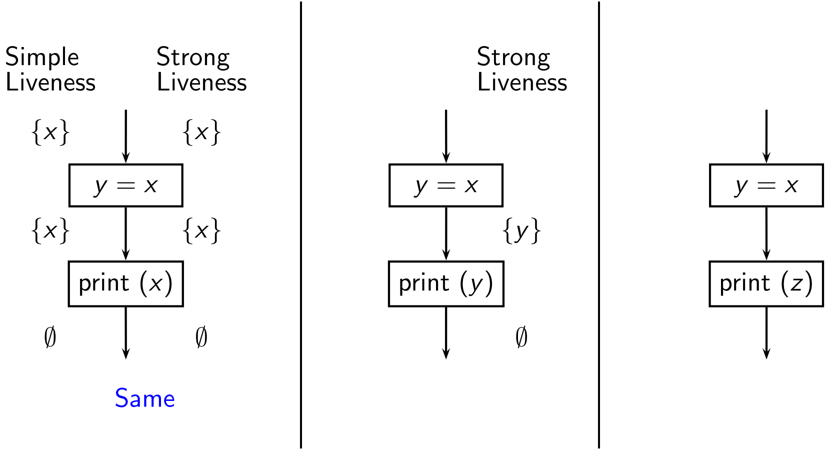
Understanding Strong Liveness



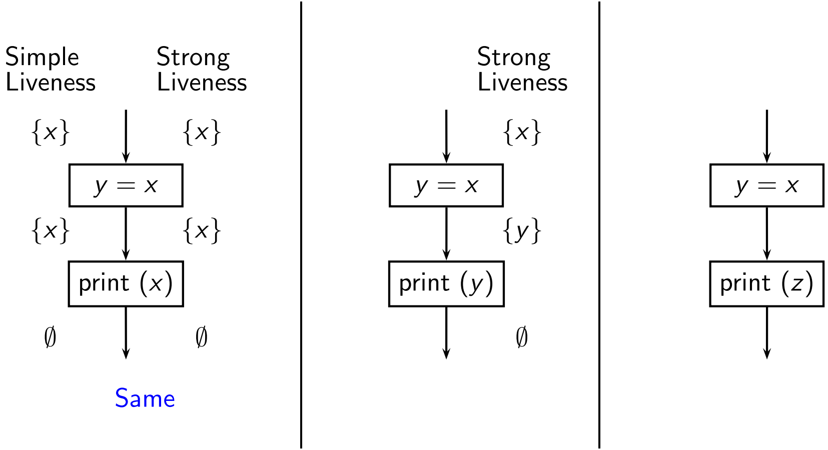




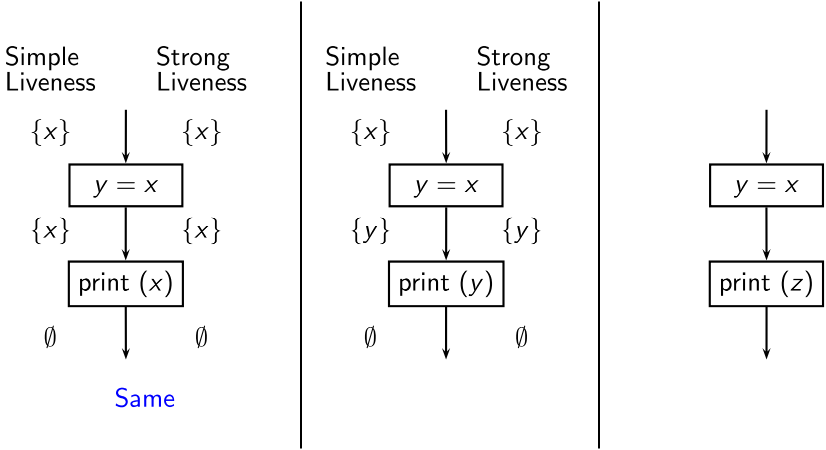
Understanding Strong Liveness



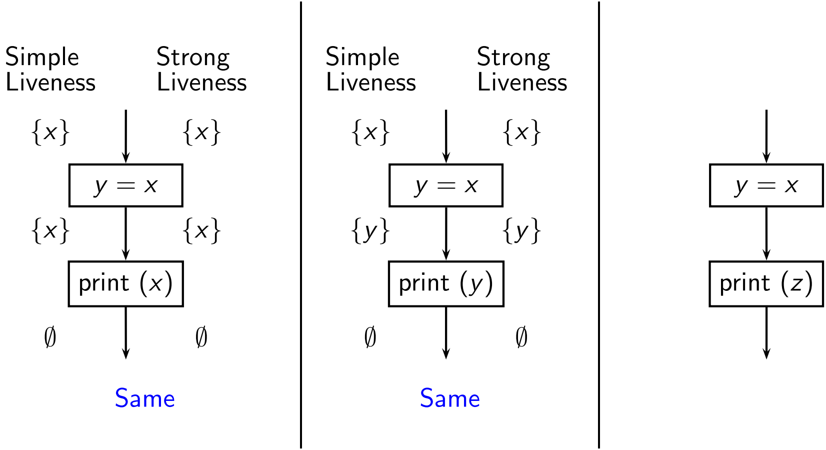
Understanding Strong Liveness



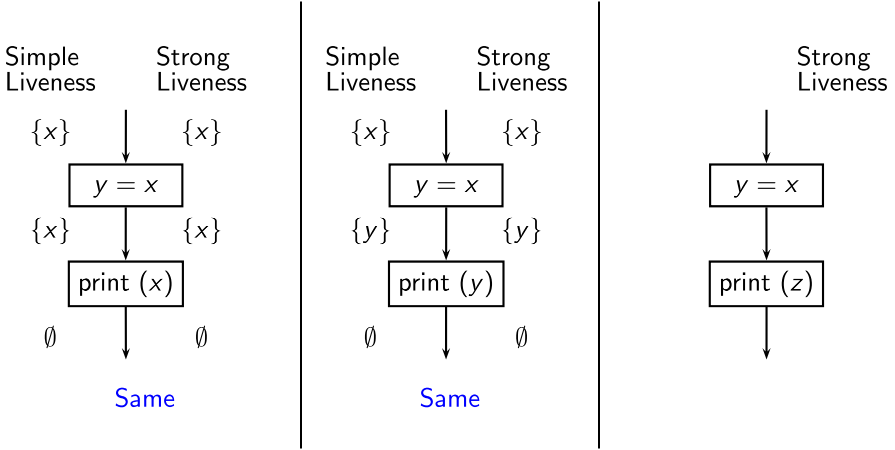
Understanding Strong Liveness



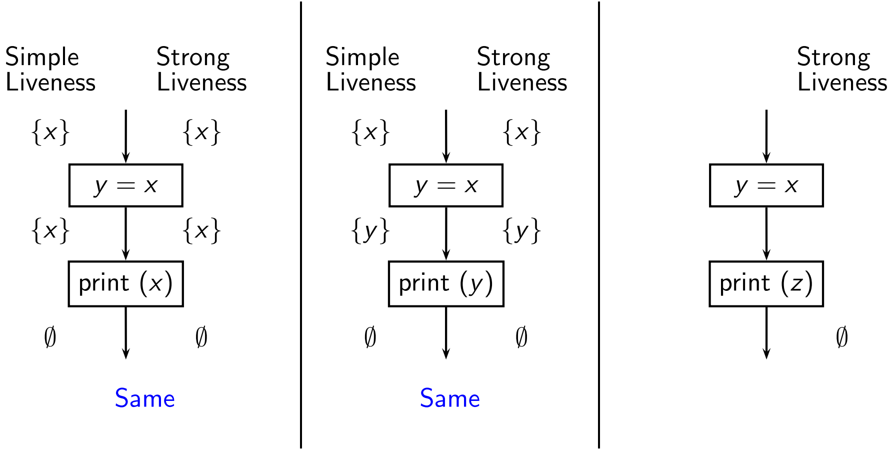
Understanding Strong Liveness



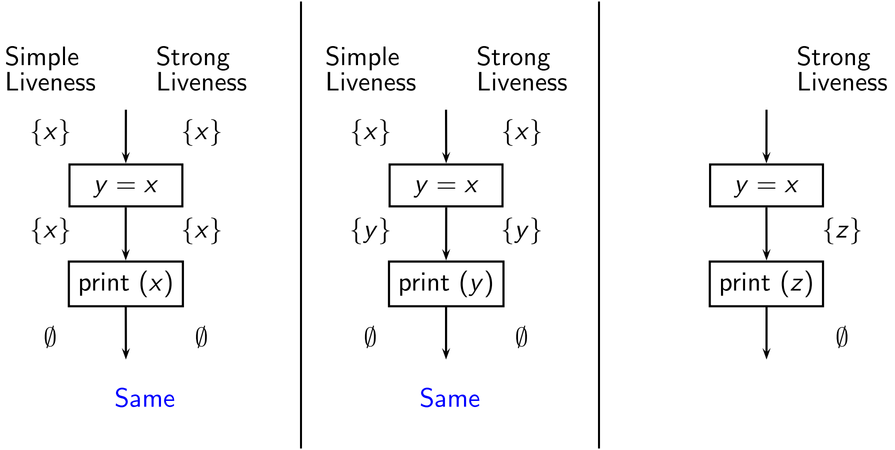
Understanding Strong Liveness



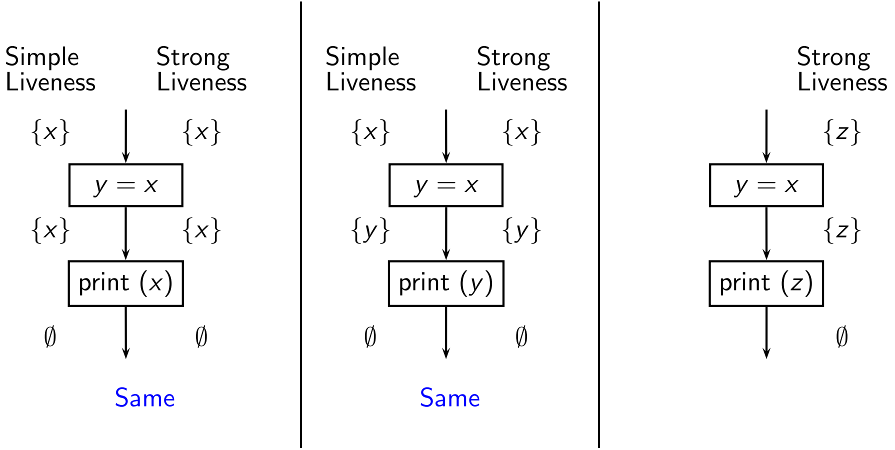
Understanding Strong Liveness



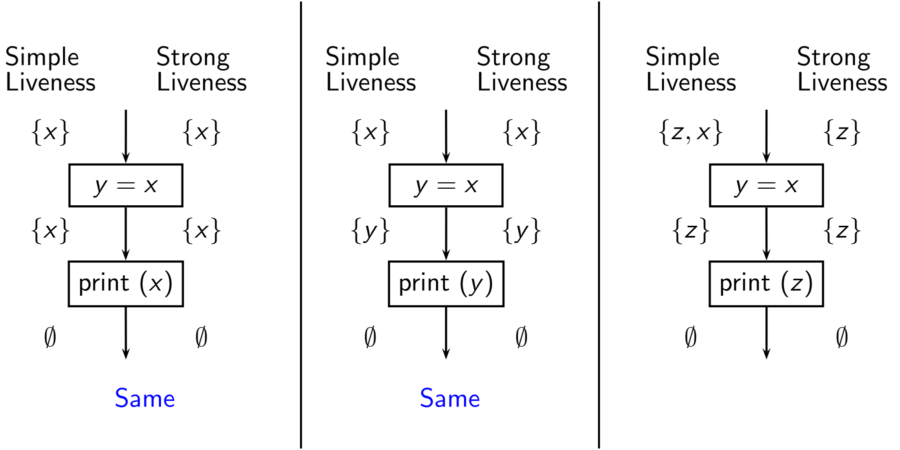
Understanding Strong Liveness



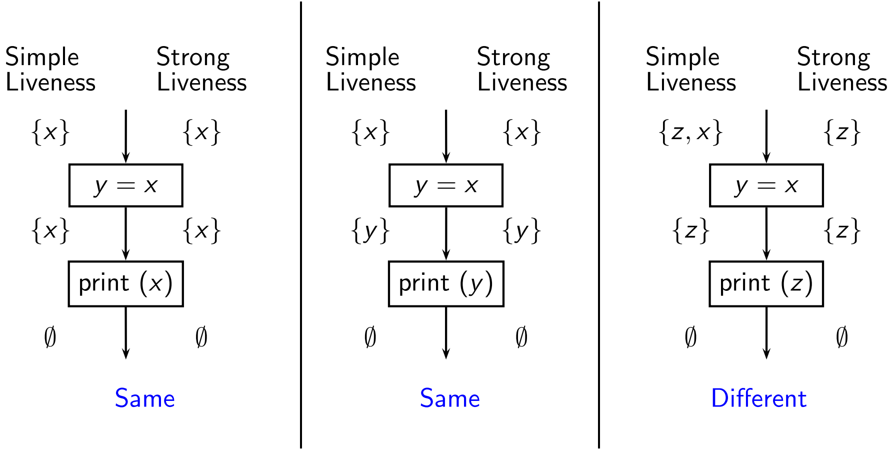
Understanding Strong Liveness



Understanding Strong Liveness



Understanding Strong Liveness



Live Variables Analysis: Simple and Strong Liveness

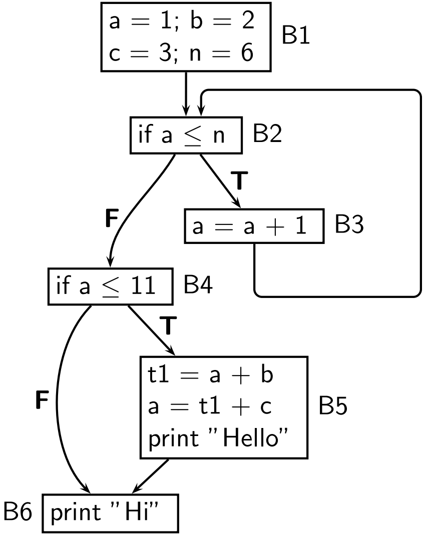
- A variable is live at a program point if its current value is likely to be used later

Live Variables Analysis: Simple and Strong Liveness

- A variable is live at a program point if its current value is likely to be used later
- We want to compute the smallest set of variables that are live

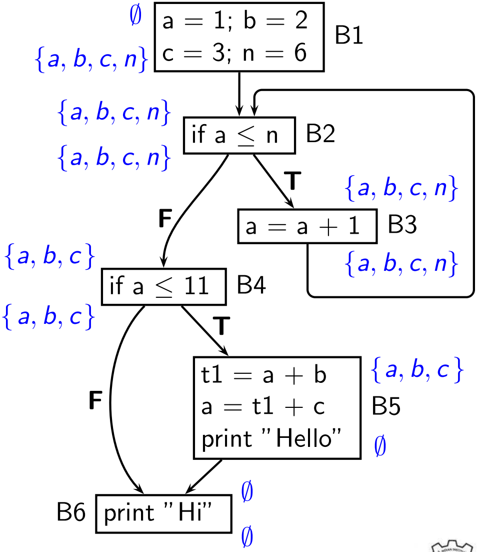
Live Variables Analysis: Simple and Strong Liveness

- A variable is live at a program point if its current value is likely to be used later
- We want to compute the smallest set of variables that are live



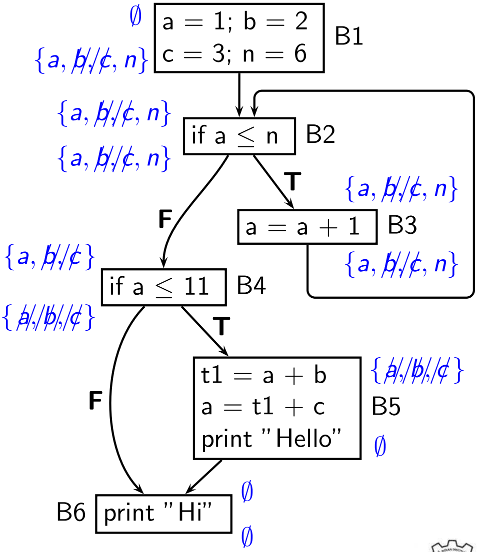
Live Variables Analysis: Simple and Strong Liveness

- A variable is live at a program point if its current value is likely to be used later
- We want to compute the smallest set of variables that are live
- Simple liveness considers every use of a variable as useful



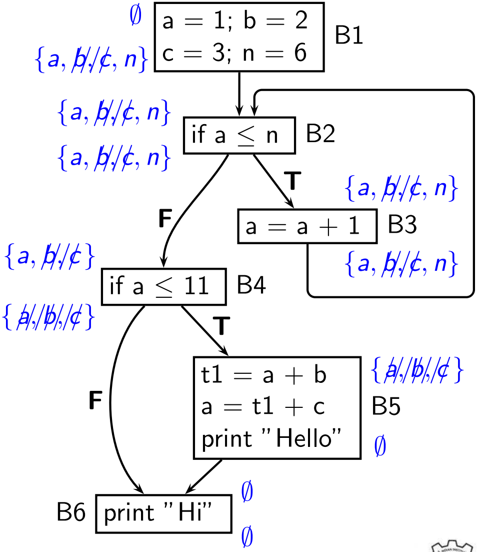
Live Variables Analysis: Simple and Strong Liveness

- A variable is live at a program point if its current value is likely to be used later
- We want to compute the smallest set of variables that are live
- Simple liveness considers every use of a variable as useful
- Strong liveness checks the liveness of the result before declaring the operands to be live



Live Variables Analysis: Simple and Strong Liveness

- A variable is live at a program point if its current value is likely to be used later
- We want to compute the smallest set of variables that are live
- Simple liveness considers every use of a variable as useful
- Strong liveness checks the liveness of the result before declaring the operands to be live
- Strong liveness is more precise than simple liveness



Data Flow Equations for Strongly Live Variables Analysis

$$
\begin{aligned}
In_n &= f_n(Out_n) \\
Out_n &= \begin{cases} BI & n \text{ is } End \\ \bigcup\limits_{s \in succ(n)} In_s & \text{otherwise} \end{cases}
\end{aligned}
$$

where,

$$
f_n(X) = \begin{cases}
(X - \{y\}) \cup (Opd(e) \cap \mathbb{V}\text{ar}) & n \text{ is } y = e, e \in \mathbb{E}\text{xpr},\ \ y \in X \\
X - \{y\} & n \text{ is } input(y) \\
X \cup \{y\} & n \text{ is } use(y) \\
X & \text{otherwise}
\end{cases}
$$

Data Flow Equations for Strongly Live Variables Analysis

$$In_n = f_n(Out_n)$$

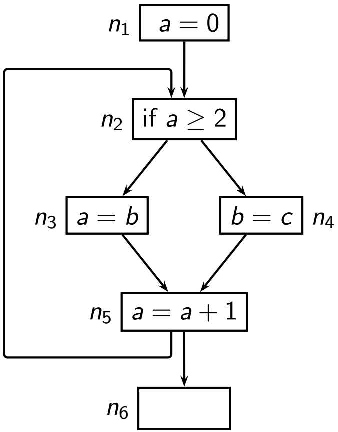
$$Out_n = \begin{cases} BI & n \text{ is } End \\ \bigcup\limits_{s \in succ(n)} In_s & \text{otherwise} \end{cases}$$

where,

$$f_n(X) = \begin{cases} (X - \{y\}) \cup (Opd(e) \cap \mathbb{V}\text{ar}) & n \text{ is } y = e, e \in \mathbb{E}\text{xpr}, \ y \in X \\ X - \{y\} & n \text{ is } input(y) \\ X \cup \{y\} & n \text{ is } use(y) \\ X & \text{otherwise} \end{cases}$$

If y is not strongly live, the assignment is skipped using the "otherwise" clause

Tutorial Problem for strongly Live Variables Analysis



Result of Strongly Live Variables Analysis

Node	Iteration #1		Iteration #2		Iteration #3		Iteration #4	
	Out_n	In_n	Out_n	In_n	Out_n	In_n	Out_n	In_n
n_6	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
n_5	$\emptyset$	$\emptyset$	$\{a\}$	$\{a\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b, c\}$	$\{a, b, c\}$
n_4	$\emptyset$	$\emptyset$	$\{a\}$	$\{a\}$	$\{a, b\}$	$\{a, c\}$	$\{a, b, c\}$	$\{a, c\}$
n_3	$\emptyset$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a, b\}$	$\{b\}$	$\{a, b, c\}$	$\{b, c\}$
n_2	$\emptyset$	$\{a\}$	$\{a, b\}$	$\{a, b\}$	$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, c\}$	$\{a, b, c\}$
n_1	$\{a\}$	$\emptyset$	$\{a, b\}$	$\{b\}$	$\{a, b, c\}$	$\{b, c\}$	$\{a, b, c\}$	$\{b, c\}$