



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

## **Data flow analysis instances**

# Where we were

- We have gone through Live Variables
  - by intuition, and
  - as a set of constraint equations that converge to a fixed point when you solve them iteratively, while
  - looking at how the solutions correspond to positions in a latticeand we've argued that the general method works for different types of information as well.
- Today, we'll try it out with
  - A few different types of elements in the sets
  - Different constraints



# The framework ingredients

- *A domain* of things to analyze
  - Sets of variables (Liveness)
  - Sets of copies
  - Sets of expressions
  - Sets of variable definitions...
- *A transfer function*
  - Gives a forward/backward direction
  - Says how to change the sets based on the program logic
- *A meet operator*
  - Says what to do when control flow paths collide

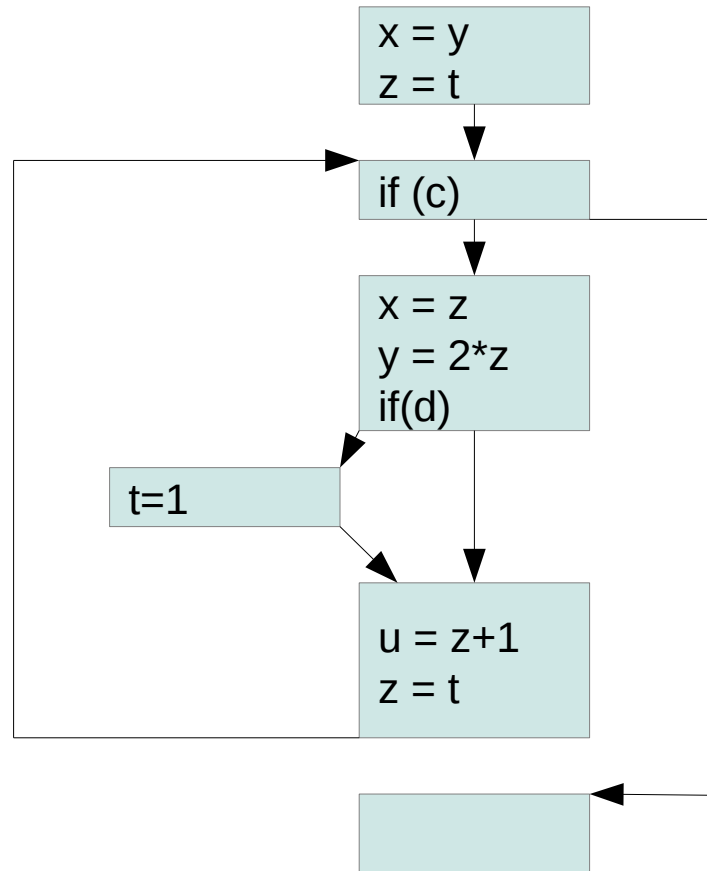


# Copy propagation

- Some variables can be copies of each other, let us detect them
  - Liveness is a backward analysis that adds set elements from any path to a program point
  - Copy propagation is a forward analysis that restricts set elements to those that are valid along every path to a program point
- We can work copy propagation out by intuition as well, to illustrate the effects of direction and choice of meet operator

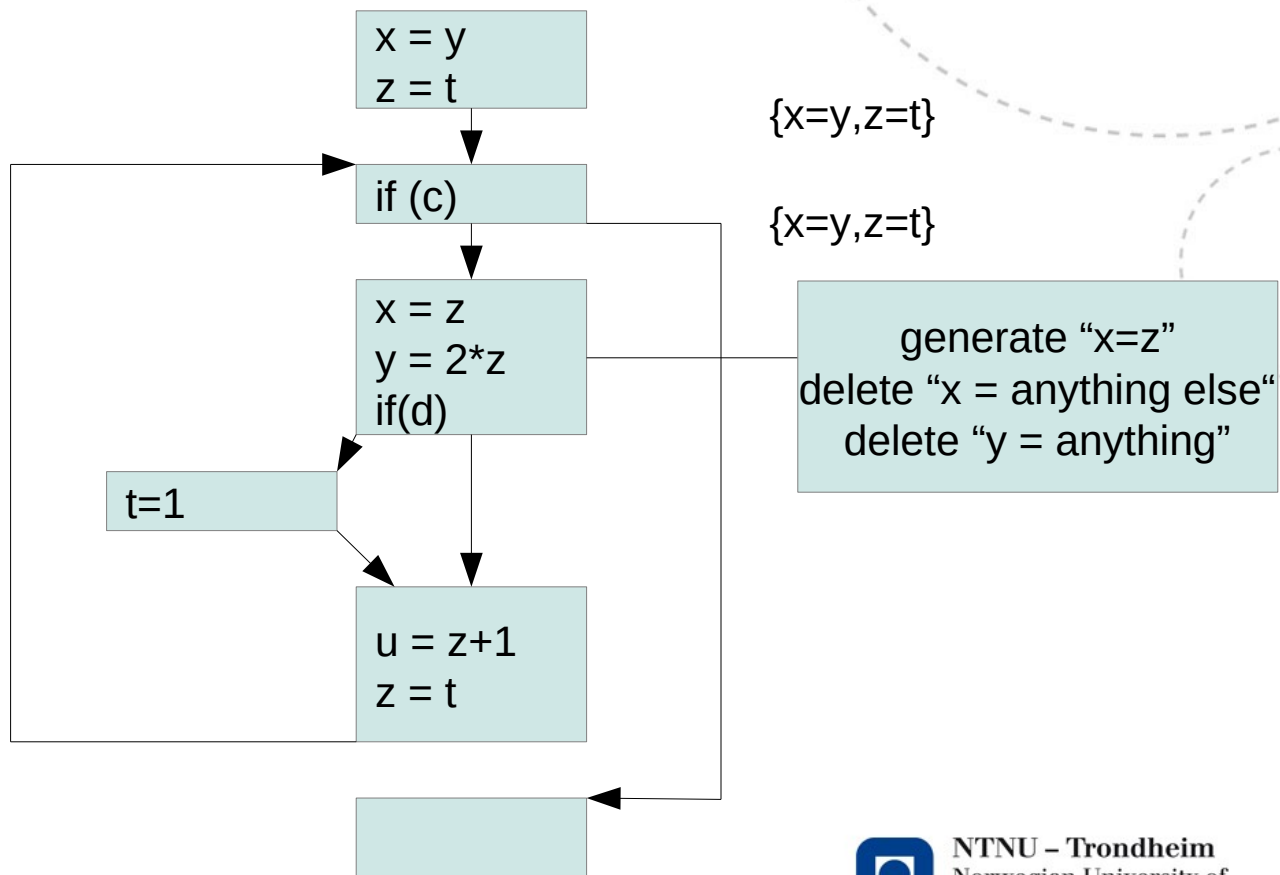
# Copy propagation

- I've modified the running example a bit, so that there are some copies to detect



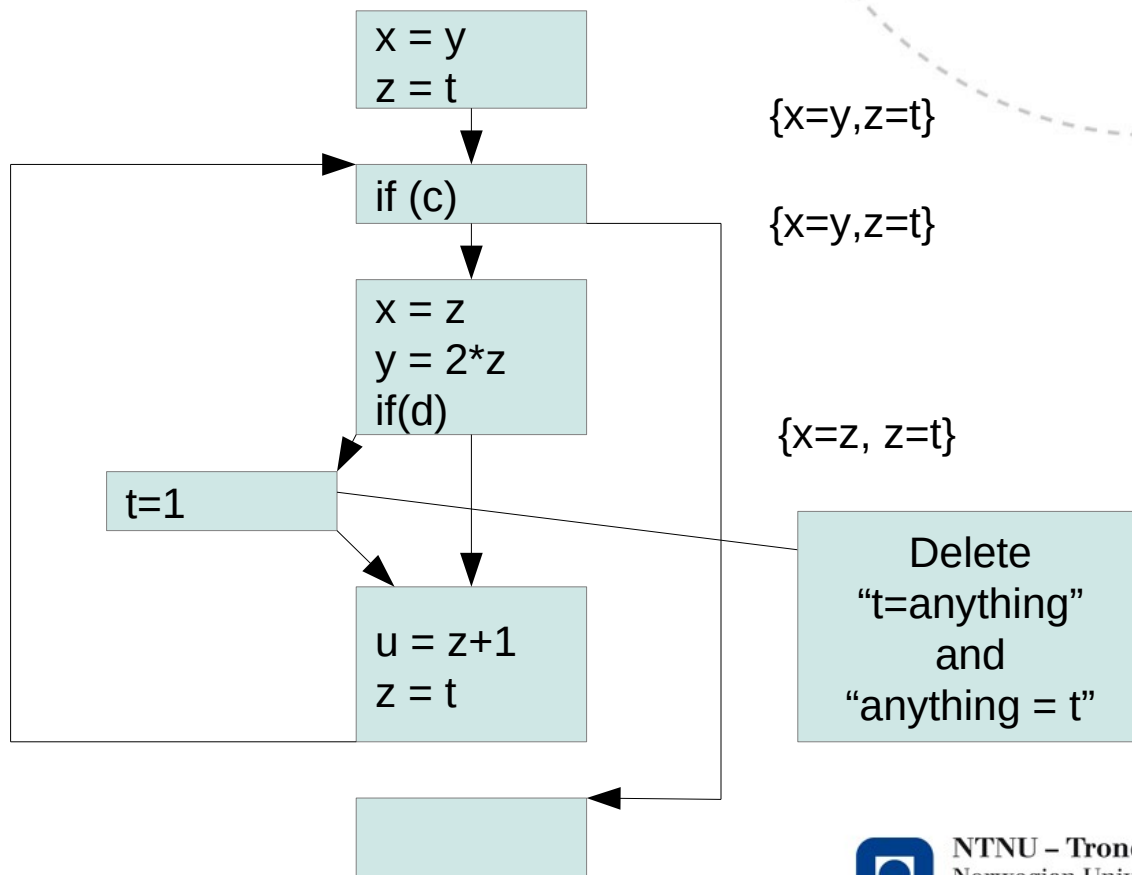
# Copy propagation

- The first two statements create some copies to start with
- The loop body makes x a copy of another variable than it was before



# Copy propagation

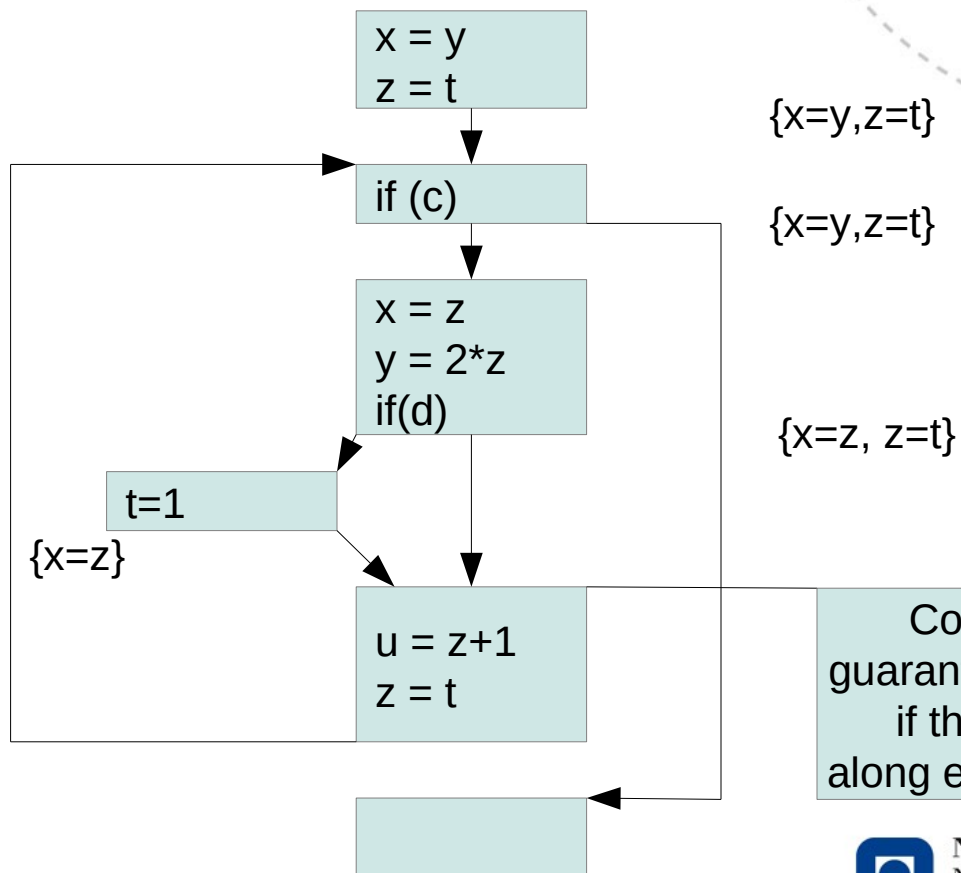
- Here's an assignment which stops  $t$  from being a copy of any other variable



# Copy propagation

- Control flows meet:

To be sure that we can treat two variables as copies of each other, there can't be any possibility that they're different



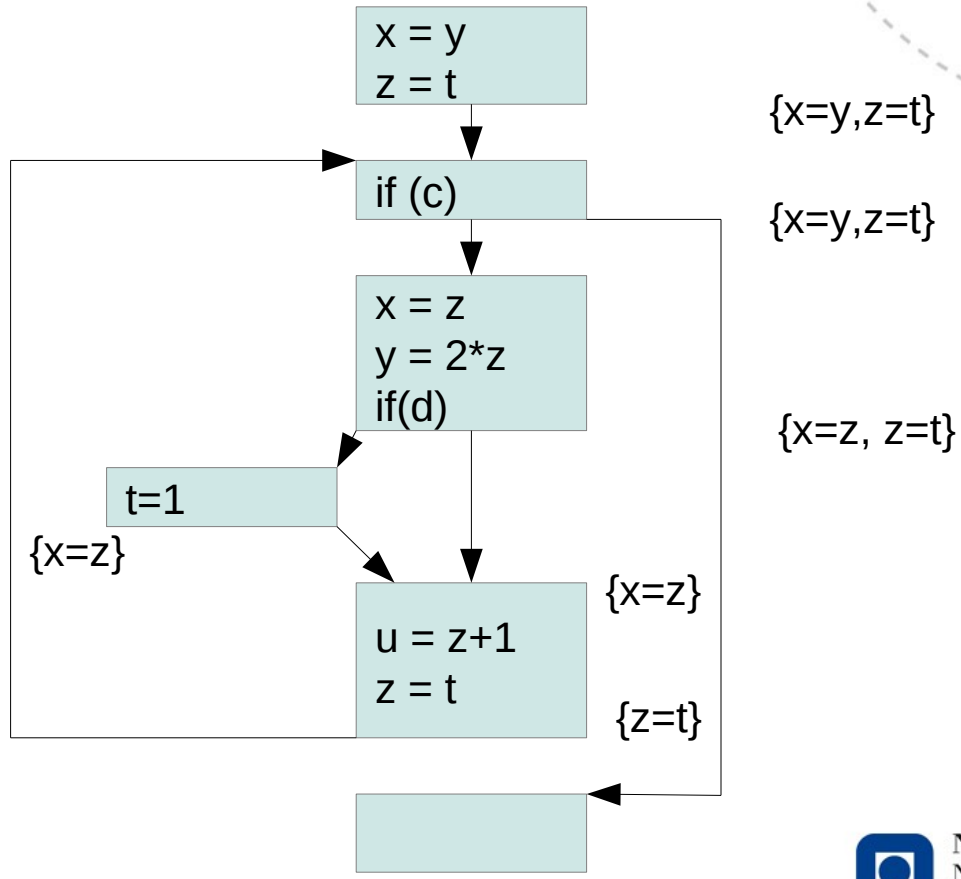
Copies are only guaranteed to be copies if they are copies along every path to here





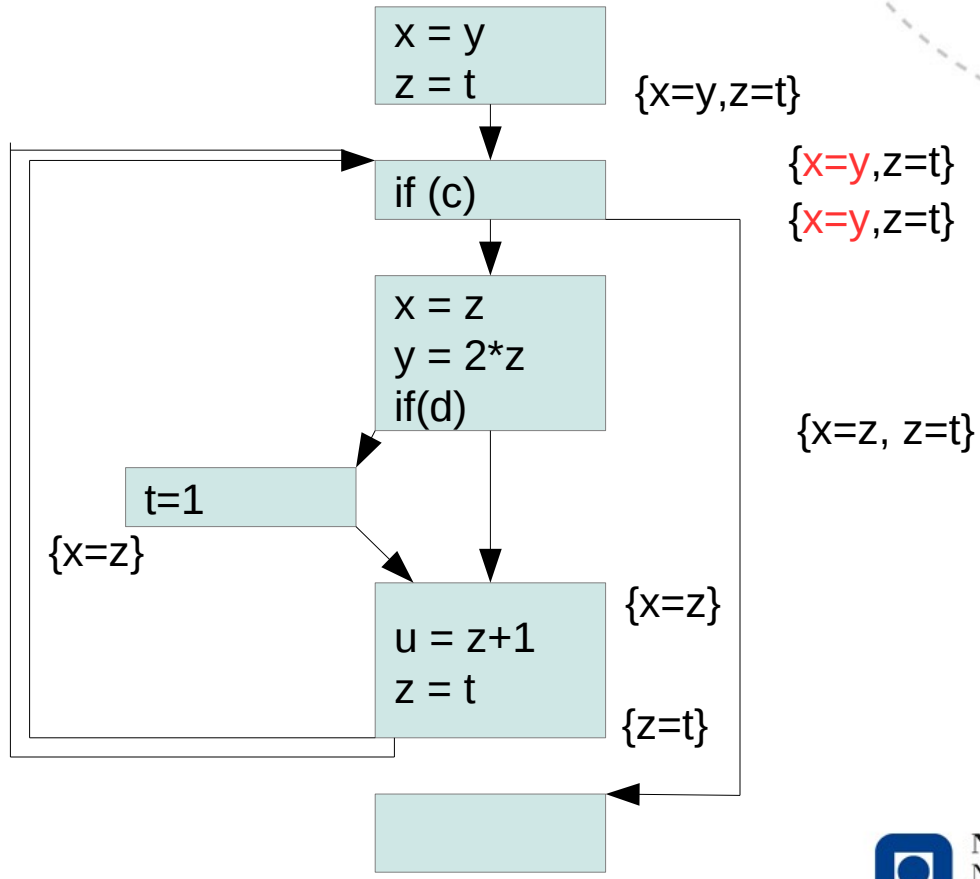


# Copy propagation

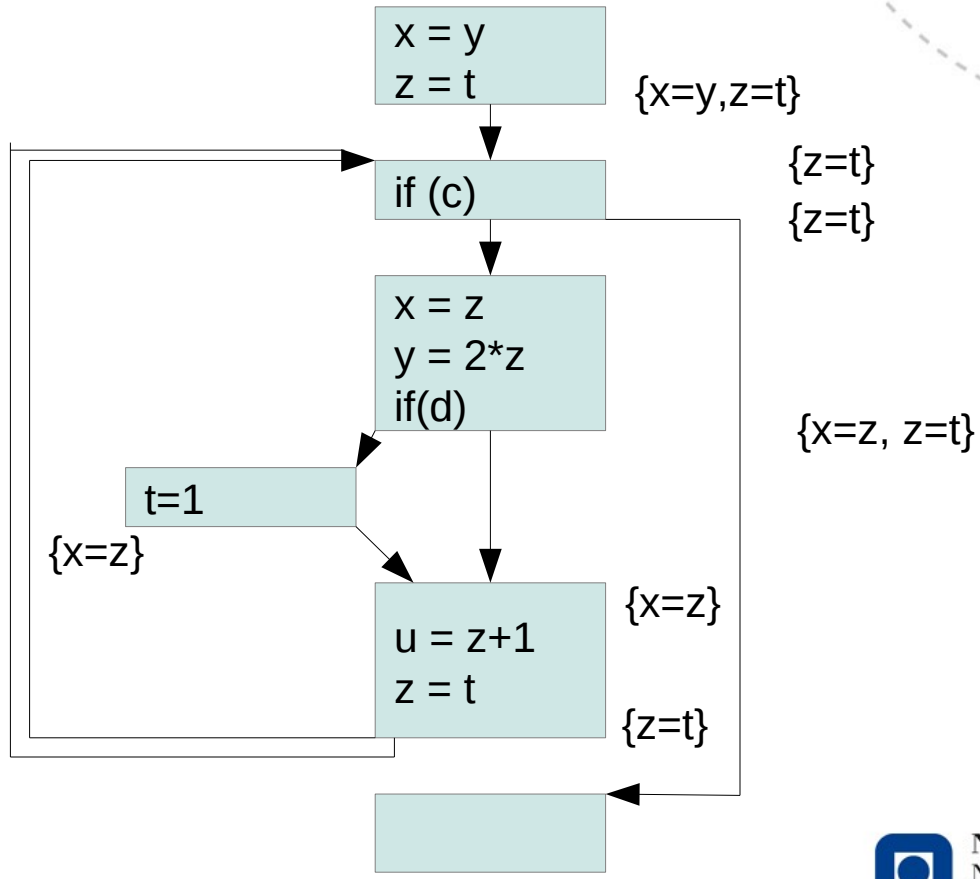


# Iteration 2

- We've found a path to the loop head where  $x=y$  is not necessarily true
- Take it away:  
 $\{x=y, z=t\} \cap \{z=t\}$   
 $= \{z=t\}$



# Solution



# Available Expressions

- An *available expression* is an expression evaluated in all program executions, which would have the same value if re-evaluated
- The sets we look for are sets of expressions, so we need to number all of those



# Available Expressions

## Data flow equations

- Instructions:

$$\text{out}[B] = F_B ( \text{in}[B] ) \quad (\text{Forward analysis})$$

- Control flow:

$$\text{in}[B] = \bigcap \{ \text{out}[B'] \mid B' \in \text{pred}(B) \} \quad (\text{Meet op. is intersection})$$

- Interpretation:

*An expression is available at the entry of B if it is available at the exit of all predecessor blocks*



# Available Expressions

## Transfer function

$$F_l(X) = \{ X - \text{kill} [l] \} \cup \text{gen} [l]$$

where

$\text{kill} [l]$  = expressions “killed” by  $l$

$\text{gen} [l]$  = expressions “generated” by  $l$

$x = y \text{ OP } z$

generates  $\{y \text{ OP } z\}$ , kills expr. with  $x$  in them

$x = \text{OP } y$

generates  $\{\text{OP } y\}$ , kills expr. with  $x$

$x = y$

generates nothing, kills expr. with  $x$

$x = \& y$

generates nothing, kills expr. with  $x$

$\text{if} (x)$

generates nothing, kills nothing

$\text{return } x$

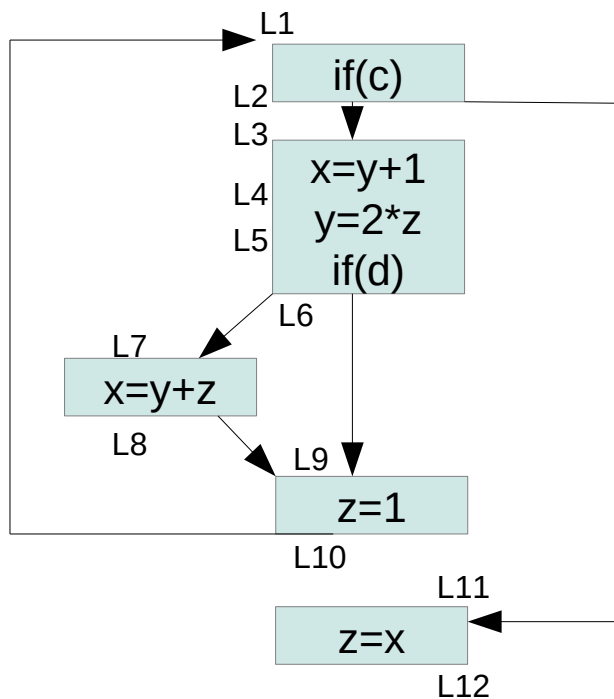
generates nothing, kills nothing

$x = f (y_1, y_2, \dots, y_n)$

generates nothing, kills expr with  $x$



# Expressions in the example



e1:  $y + 1$

e2:  $2 * z$

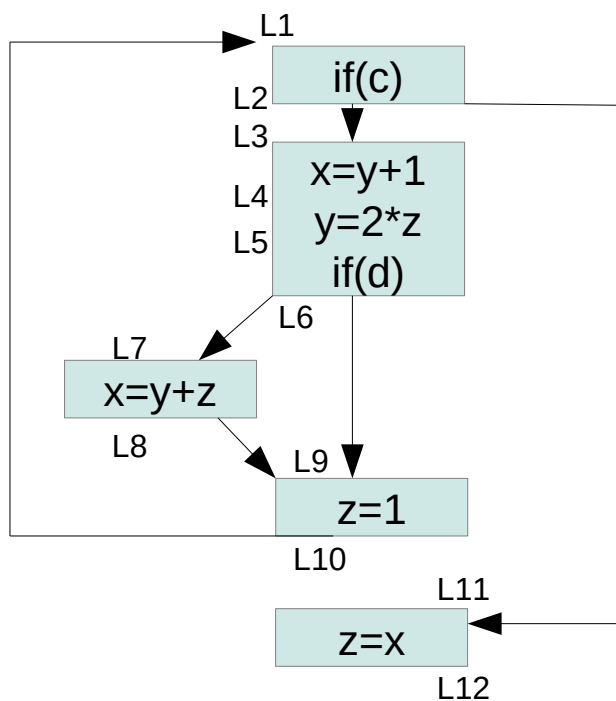
e3:  $y + z$





e1:  $y + 1$   
 e2:  $2 * z$   
 e3:  $y + z$

# Data flow equations



$$L4 = \{ L3 \} \cup \{ e1 \}$$

$$L5 = \{ L4 - e1 \} \cup e2$$

$$L8 = \{ L7 \} \cup e3$$

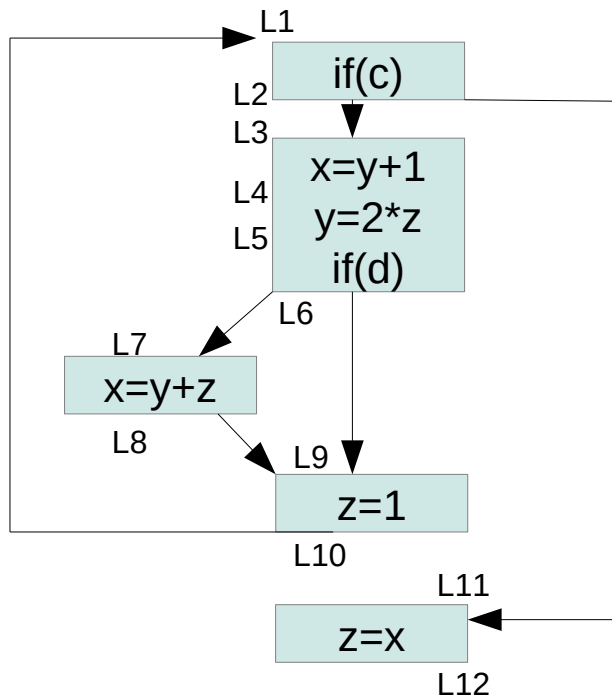
$$L9 = L6 \cap L8$$

$$L10 = L9 - \{ e2, e3 \}$$

$$L12 = L11 - \{ e2, e3 \}$$

# Iteration 1

e1:  $y + 1$   
 e2:  $2 * z$   
 e3:  $y + z$



L1 =

L2 =

L3 =

L4 = e1

L5 =  $\{e1 - e1\} \cup e2 = e2$

L6 = e2

L7 = e2

L8 = e2, e3

L9 =  $\{e2\} \cap \{e2, e3\} = e2$

L10 =  $\{e2 - e2\} = \{\}$

L11 =

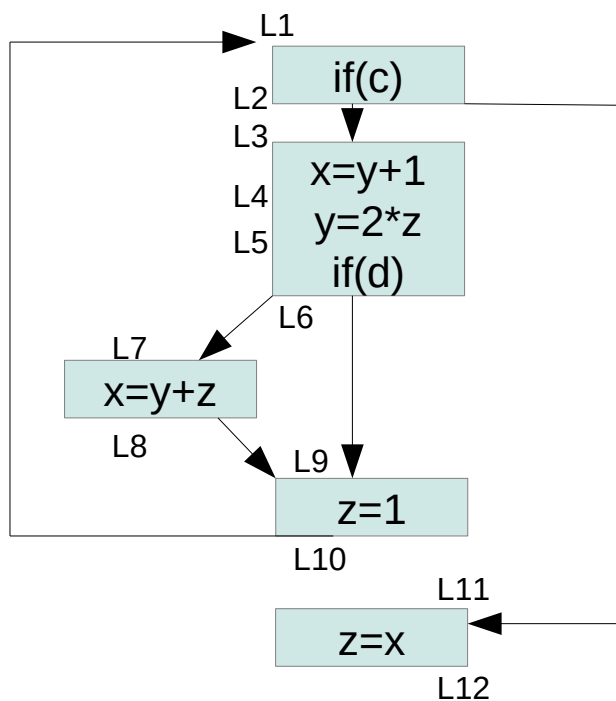
L12 =

L1 =  $\{\} \cap L10$   
 L4 =  $\{L3\} \cup \{e1\}$   
 L5 =  $\{L4 - e1\} \cup e2$   
 L8 =  $\{L7\} \cup e3$   
 L9 =  $L6 \cap L8$   
 L10 =  $L9 - \{e2, e3\}$   
 L11 = L1  
 L12 =  $L11 - \{e2, e3\}$



# Iteration 2: no change

e1:  $y + 1$   
 e2:  $2 * z$   
 e3:  $y + z$



L1 =  
 L2 =  
 L3 =  
 L4 = e1  
 L5 = e2  
 L6 = e2  
 L7 = e2  
 L8 = e2,e3  
 L9 = e2  
 L10 =  
 L11 =  
 L12 =

L1 =  $\{\} \cap L10$   
 L4 =  $\{L3\} \cup \{e1\}$   
 L5 =  $\{L4 - e1\} \cup e2$   
 L8 =  $\{L7\} \cup e3$   
 L9 =  $L6 \cap L8$   
 L10 =  $L9 - \{e2, e3\}$   
 L11 = L1  
 L12 =  $L11 - \{e2, e3\}$



# Reaching Definitions

- A *reaching definition* is a definition of a variable where the assigned value *may* be observed at a program point in some execution
- The sets we look for are sets of assignments, so we need to number all of those



# Reaching Definitions

## Data flow equations

- Instructions:

$$\text{out}[B] = F_B ( \text{in}[B] ) \quad (\text{Forward analysis})$$

- Control flow:

$$\text{in}[B] = \bigcup \{ \text{out}[B'] \mid B' \in \text{pred}(B) \} \quad (\text{Meet op. is union})$$

- Interpretation:

*A definition reaches the entry of block B if it reaches the exit of at least one of its predecessor nodes*

# Reaching Definitions

## Transfer function

$$F_l(X) = \{ X - \text{kill } [l] \} \cup \text{gen } [l]$$

where

kill [ l ] = definitions “killed” by l

gen [ l ] = definitions “generated” by l

$x = y \text{ OP } z$

generates  $\{x = y \text{ OP } z\}$ , kills other definitions of x

$x = \text{OP } y$

generates  $\{x = \text{OP } y\}$ , kills other definitions of x

$x = y$

generates  $\{x = y\}$ , kills other definitions of x

$x = \& y$

generates  $\{x = \&y\}$ , kills other definitions of x

if ( x )

generates nothing, kills nothing

return x

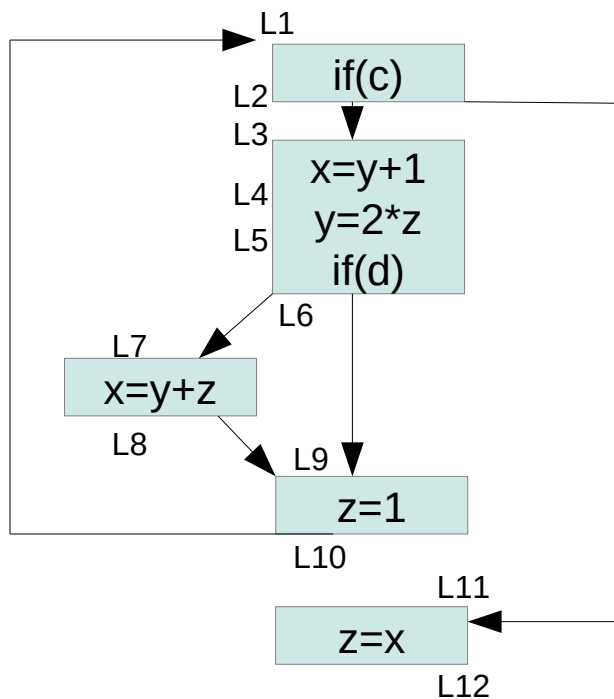
generates nothing, kills nothing

$x = f ( y_1, y_2, \dots, y_n )$

generates  $\{x = f\dots\}$ , kills other definitions of x



# Definitions in the example



d1:  $x = y + 1$

d2:  $y = 2 * z$

d3:  $x = y + z$

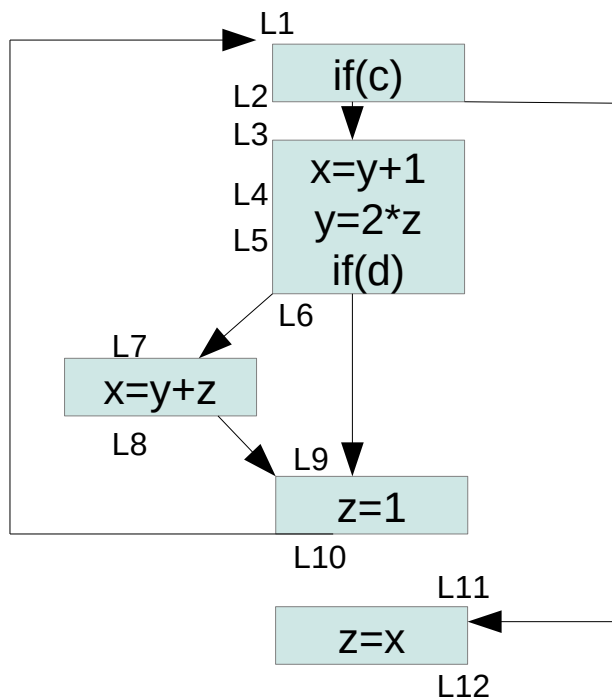
d4:  $z = 1$

d5:  $z = x$



d1:  $x = y+1$   
 d2:  $y = 2*z$   
 d3:  $x = y+z$   
 d4:  $z = 1$   
 d5:  $z = x$

# Data flow equations



$$L1 = \{\} \cup L10$$

$$L4 = \{L3-d3\} \cup d1$$

$$L5 = L4 \cup d2$$

$$L8 = \{L7-d1\} \cup d3$$

$$L9 = L6 \cup L8$$

$$L10 = \{L9-d5\} \cup d4$$

$$L11 = L2$$

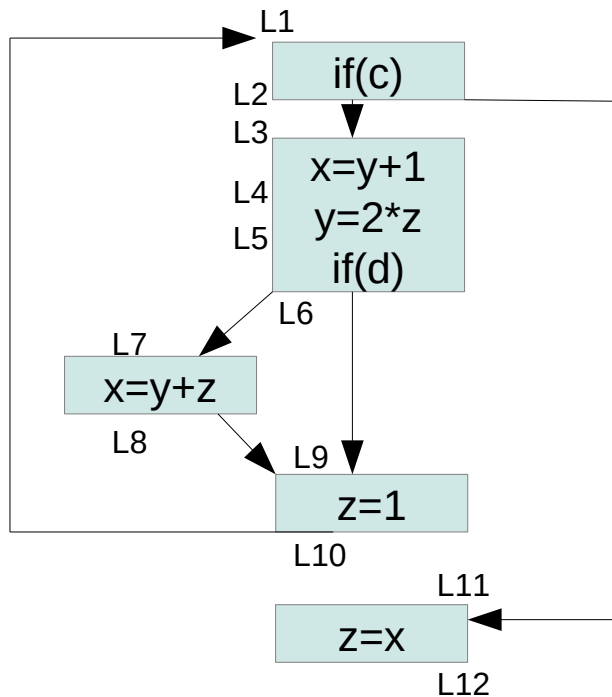
$$L12 = \{L11-d4\} \cup d5$$





# Iteration 1

d1:  $x = y + 1$   
 d2:  $y = 2 * z$   
 d3:  $x = y + z$   
 d4:  $z = 1$   
 d5:  $z = x$



L1 =

L2 =

L3 =

L4 = d1

L5 = d1,d2

L6 = d1,d2

L7 = d1,d2

L8 = d2,d3

L9 =  $\{d1,d2\} \cup \{d2,d3\} = \{d1,d2,d3\}$

L10 =  $\{d1,d2,d3\} \cup d4 = \{d1,d2,d3,d4\}$

L11 =

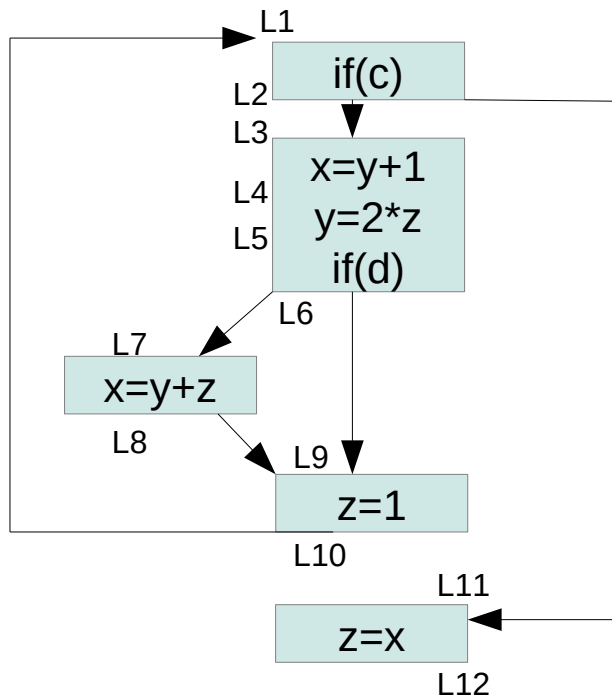
L12 = d5

L1 =  $\{\} \cup L10$   
 L4 =  $\{L3-d3\} \cup d1$   
 L5 =  $L4 \cup d2$   
 L8 =  $\{L7-d1\} \cup d3$   
 L9 =  $L6 \cup L8$   
 L10 =  $\{L9-d5\} \cup d4$   
 L11 = L2  
 L12 =  $\{L11-d4\} \cup d5$



# Iteration 2

d1:  $x = y+1$   
 d2:  $y = 2*z$   
 d3:  $x = y+z$   
 d4:  $z = 1$   
 d5:  $z = x$



$$L1 = \{\} \cup \{d1, d2, d3, d4\}$$

$$L2 = d1, d2, d3, d4$$

$$L3 = d1, d2, d3, d4$$

$$L4 = d1, d2, d4$$

$$L5 = d1, d2, d4$$

$$L6 = d1, d2, d4$$

$$L7 = d1, d2, d4$$

$$L8 = d2, d3, d4$$

$$L9 = d1, d2, d3 \cup \{d1, d2, d4\} = \{d1, d2, d3, d4\}$$

$$L10 = d1, d2, d3, d4$$

$$L11 = d1, d2, d3, d4$$

$$L12 = d5 \cup d1, d2, d3 = \{d1, d2, d3, d5\}$$

$$L1 = \{\} \cup L10$$

$$L4 = \{L3-d3\} \cup d1$$

$$L5 = L4 \cup d2$$

$$L8 = \{L7-d1\} \cup d3$$

$$L9 = L6 \cup L8$$

$$L10 = \{L9-d5\} \cup d4$$

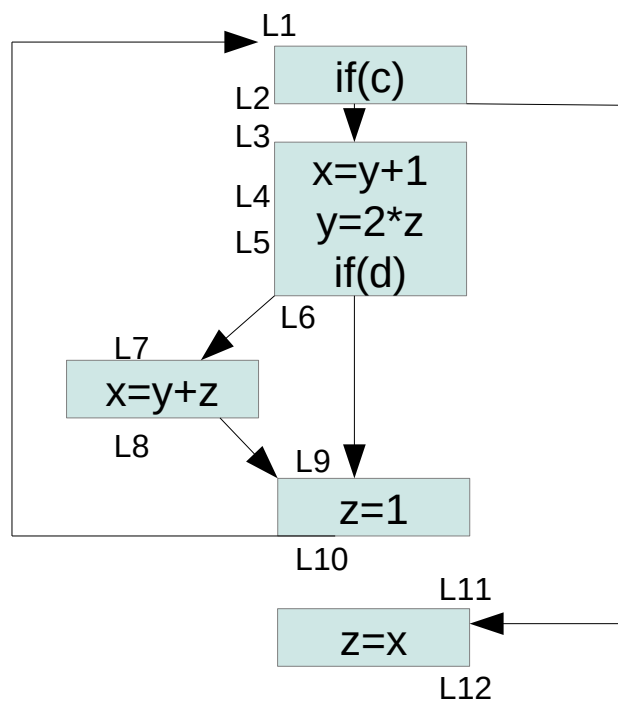
$$L11 = L2$$

$$L12 = \{L11-d4\} \cup d5$$



# Iteration 3: no change

d1:  $x = y+1$   
 d2:  $y = 2*z$   
 d3:  $x = y+z$   
 d4:  $z = 1$   
 d5:  $z = x$



L1 = d1,d2,d3,d4

L2 = d1,d2,d3,d4

L3 = d1,d2,d3,d4

L4 = d1,d2,d4

L5 = d1,d2,d4

L6 = d1,d2,d4

L7 = d1,d2,d4

L8 = d2,d3,d4

L9 = d1,d2,d3,d4

L10 = d1,d2,d3,d4

L11 = d1,d2,d3,d4

L12 = d1,d2,d3,d5

L1 =  $\{\} \cup L10$   
 L4 =  $\{L3-d3\} \cup d1$   
 L5 =  $L4 \cup d2$   
 L8 =  $\{L7-d1\} \cup d3$   
 L9 =  $L6 \cup L8$   
 L10 =  $\{L9-d5\} \cup d4$   
 L11 = L2  
 L12 =  $\{L11-d4\} \cup d5$



# Key takeaways

- The choice of domain determines what we're analyzing
- With union as meet operator, we get “maybe”-analyses
  - There is a path where an element was introduced
- With intersection as meet operator, we get “must”-analyses
  - Every path introduces these elements

# Next time

- We will
  - put all these (and one more) into a big ol' overview
  - take out the lattices again, and try to say something about how well the fixed point solution characterizes the analyzed program
  - invent a function which lets us use the same method to detect loops