NTNU – Trondheim
Norwegian University of
Science and Technology

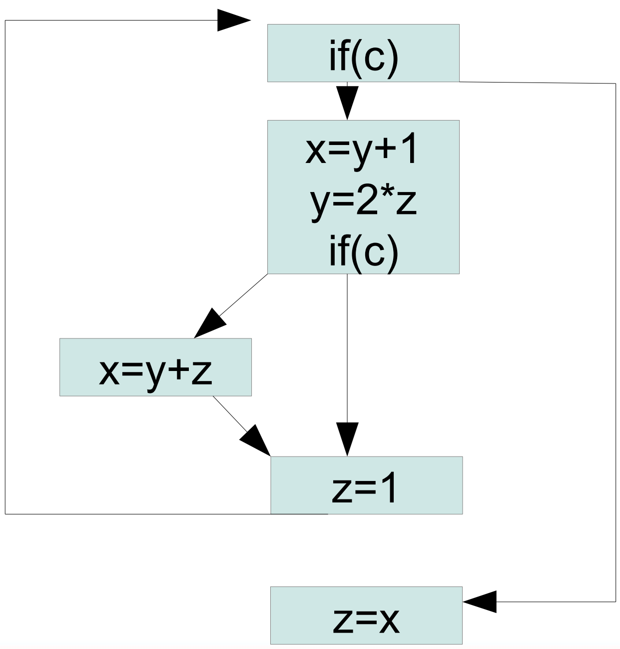
Liveness revisited

Putting “the framework” into practice

- Having introduced some ideas and notation, it might be useful to visit the liveness analysis again
- This time, we’ll apply the notation and connect it to the (somewhat abstract) argument that it works
- Thus, we can use the same ideas and notation for other analysis instances next time

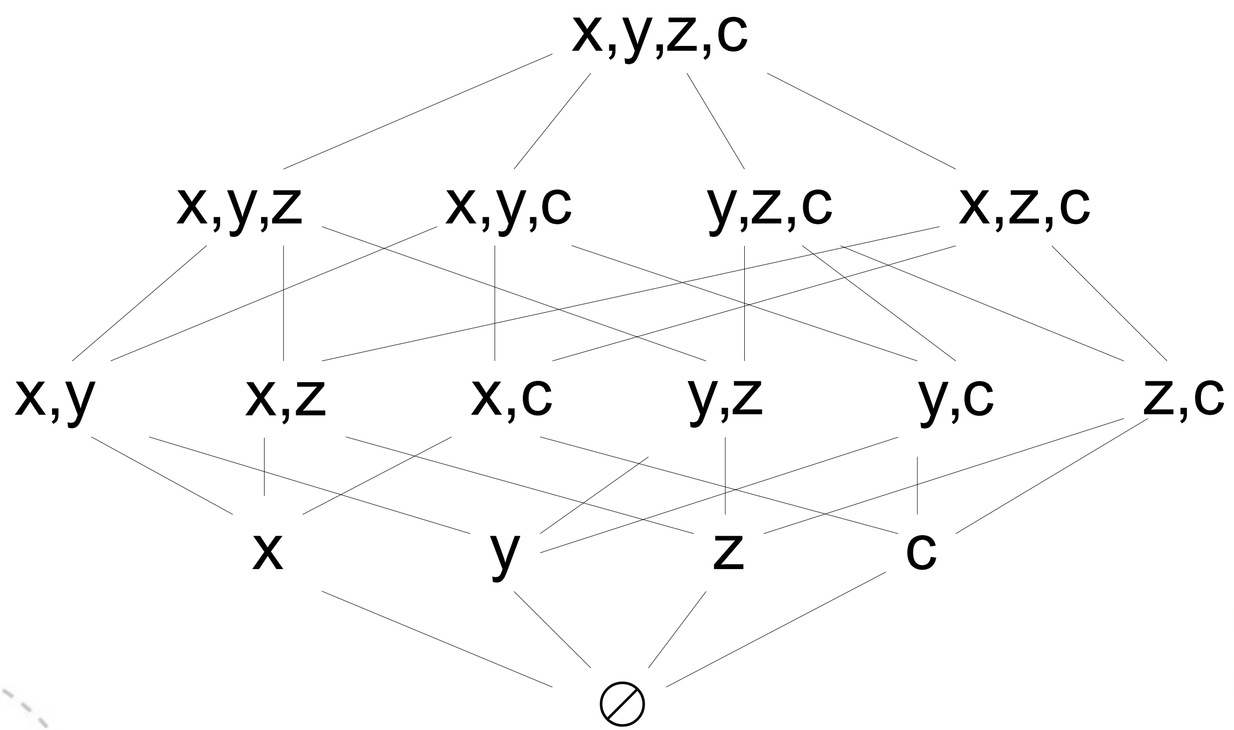
Slightly modified liveness example

- I have removed the variable ‘d’, to have fewer variables to deal with
  - This makes the program a bit stupider, but it’ll work for illustration



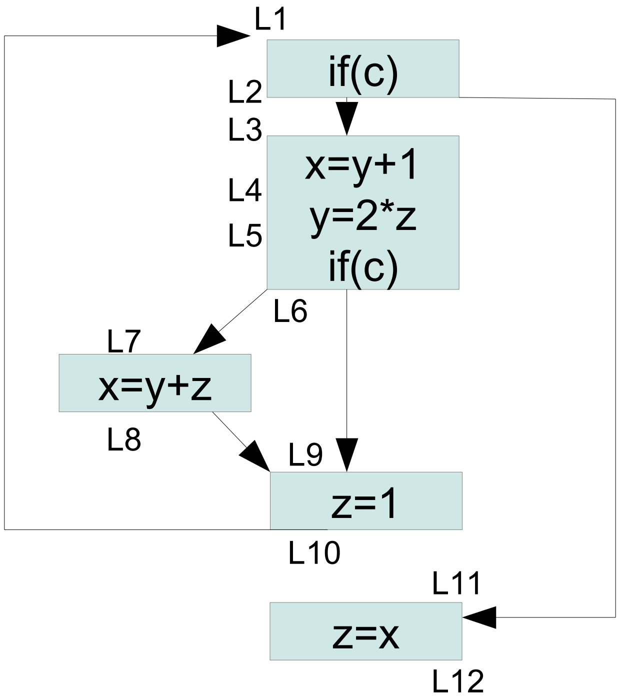
The power set lattice

- This is why I want one less variable to deal with



Name all the program points

so that we can talk about them in multiple diagrams



NTNU – Trondheim
Norwegian University of
Science and Technology

Recipe for the constraints

- Constraints from instructions:

  $in[I] = \{out[I] – def[I]\} \cup use[I]$

- Constraints from control flow:

  $out[B] = \cup\ in[B'] \mid B'$ is a successor of B

Data flow equations for each point

L1 = L2 ∪ {c}

L2 = L3 ∪ L11

L3 = {L4 – x} ∪ {y}

L4 = {L5 – y} ∪ {z}

L5 = L6 ∪ {c}

L6 = L7 ∪ L9

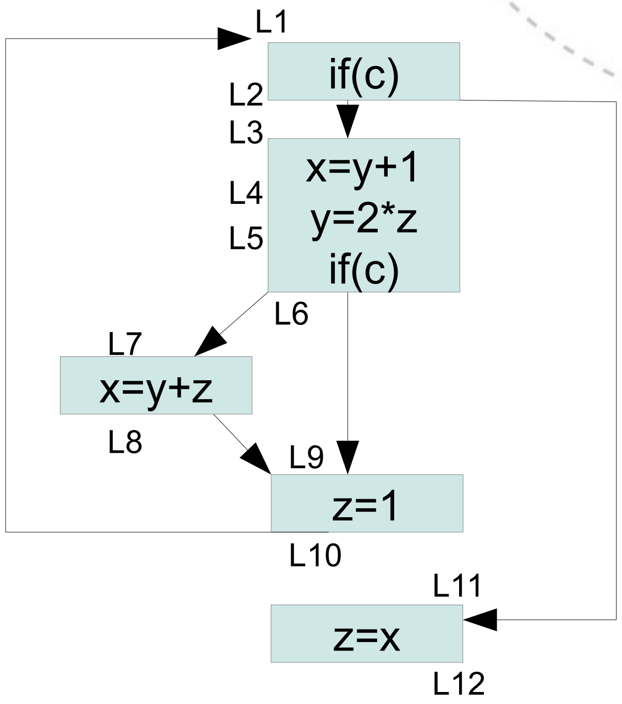
L7 = {L8 – x} ∪ {y,z}

L8 = L9

L9 = {L10 – z}

L10 = L1

L11 = {L12 – z} ∪ {x}



An initial assumption

- Last time, I took the commonsensical approach that the variables will see some future use we know nothing about
- This was a tiny fib, so as to get to the data flow thing without waving my hands around what this program ostensibly "does"
- When you're analyzing an entire function/program/translation unit, it is actually quite safe to say that nothing will be used again at the end

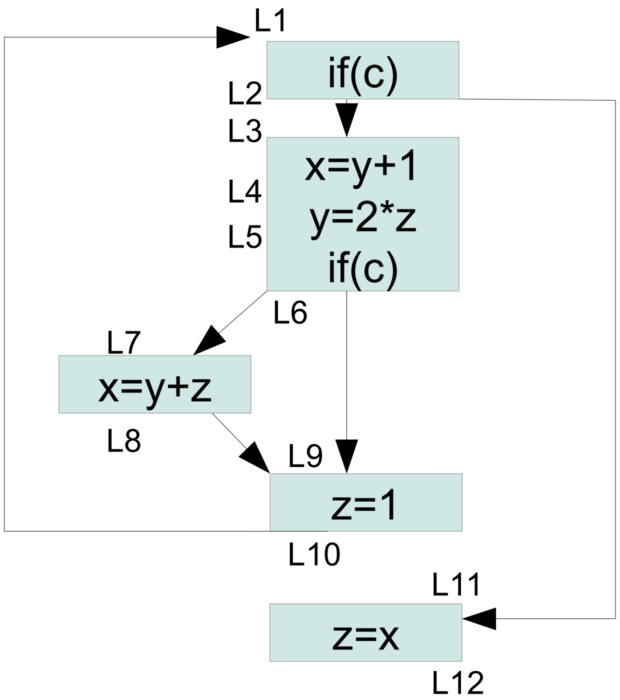
NTNU – Trondheim
Norwegian University of
Science and Technology

The handwaving

- Since we're re-doing this with all the trimmings now, please make-believe that this is an independent program unit
- That is pretty contrived
  - In the context of optimizations, the entire code should actually be cut away, it does nothing observable
  - If we can ignore that, while still pretending to be interested in the liveness result, we can work out the constraints from the more appropriate starting point of the empty set
- Yes, I know it's a bit corny to optimize pointless code
- Keeps the example small, though

Iteration 1, L11

L1 = L2 ∪ {c}
L2 = L3 ∪ L11
L3 = {L4 – x} ∪ {y}
L4 = {L5 – y} ∪ {z}
L5 = L6 ∪ {c}
L6 = L7 ∪ L9
L7 = {L8 – x} ∪ {y,z}
L8 = L9
L9 = {L10 – z}
L10 = L1
L11 = {L12 – z} ∪ {x}

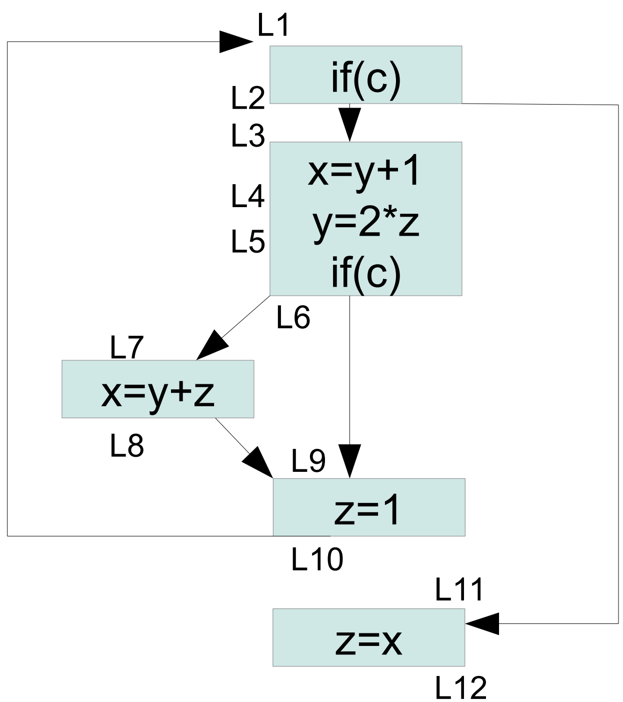


L1 = {}
L2 = {}
L3 = {}
L4 = {}
L5 = {}
L6 = {}
L7 = {}
L8 = {}
L9 = {}
L10 = {}
L11 = {x}
L12 = {}

NTNU – Trondheim
Norwegian University of
Science and Technology

Iteration 1, L10→L3

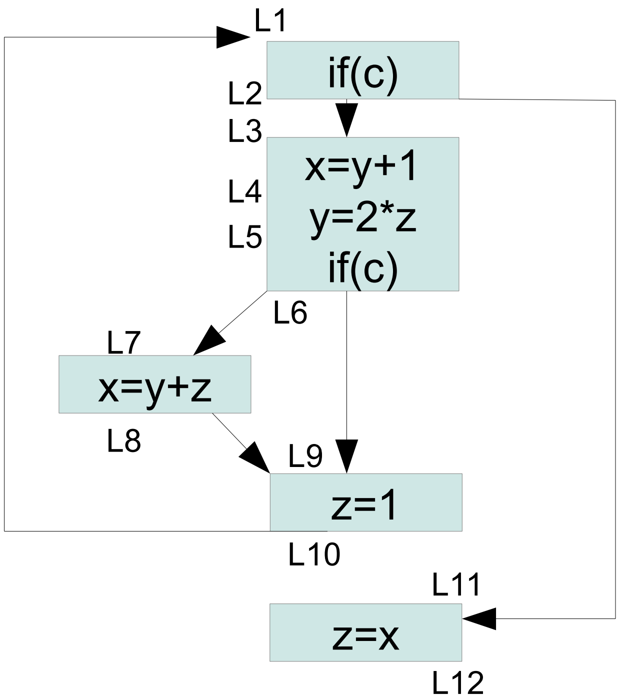
L1 = L2 ∪ {c}
L2 = L3 ∪ L11
L3 = {L4 – x} ∪ {y}
L4 = {L5 – y} ∪ {z}
L5 = L6 ∪ {c}
L6 = L7 ∪ L9
L7 = {L8 – x} ∪ {y,z}
L8 = L9
L9 = {L10 – z}
L10 = L1
L11 = {L12 – z} ∪ {x}



L1 = {}
L2 = {}
L3 = {y,z,c}
L4 = {z,c}
L5 = {y,z,c}
L6 = {y,z}
L7 = {y,z}
L8 = {}
L9 = {}
L10 = {}
L11 = {x}
L12 = {}

Iteration 1, L2,L1

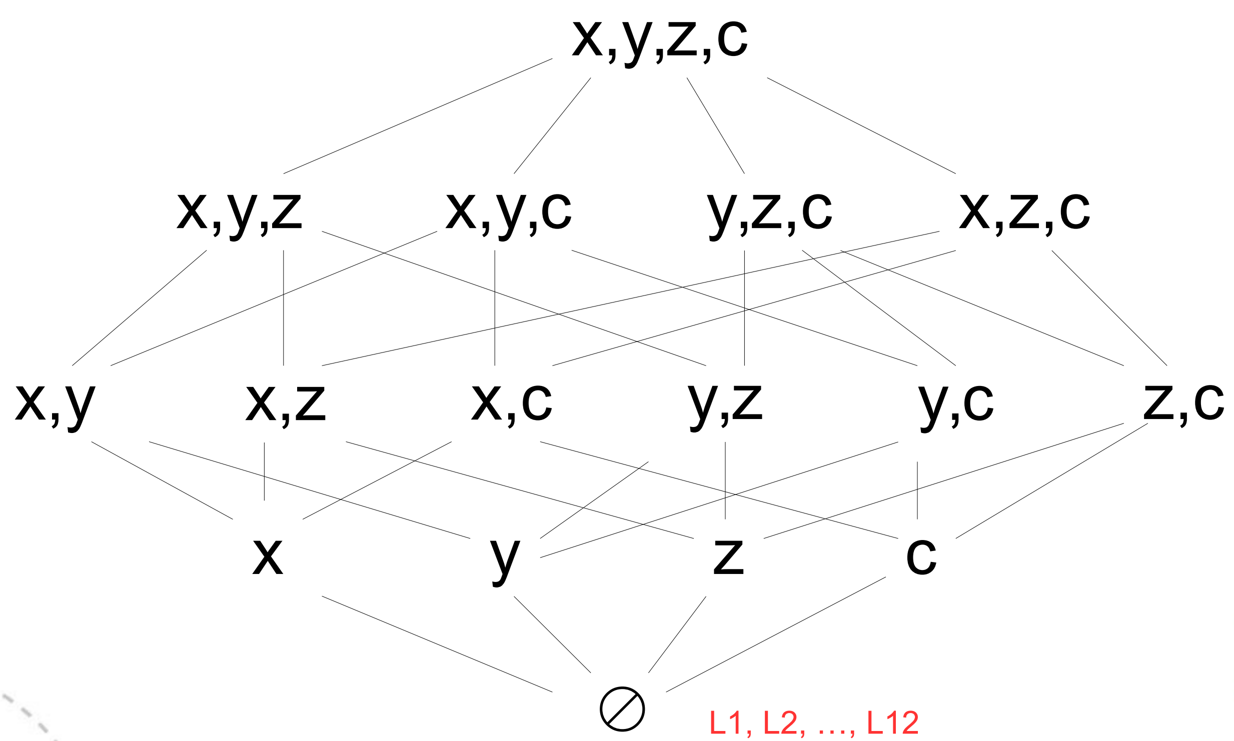
L1 = L2 ∪ {c}
L2 = L3 ∪ L11
L3 = {L4 – x} ∪ {y}
L4 = {L5 – y} ∪ {z}
L5 = L6 ∪ {c}
L6 = L7 ∪ L9
L7 = {L8 – x} ∪ {y,z}
L8 = L9
L9 = {L10 – z}
L10 = L1
L11 = {L12 – z} ∪ {x}



L1 = {x,y,z,c}
L2 = {x,y,z,c}
L3 = {y,z,c}
L4 = {z,c}
L5 = {y,z,c}
L6 = {y,z}
L7 = {y,z}
L8 = {}
L9 = {}
L10 = {}
L11 = {x}
L12 = {}

The program points have moved

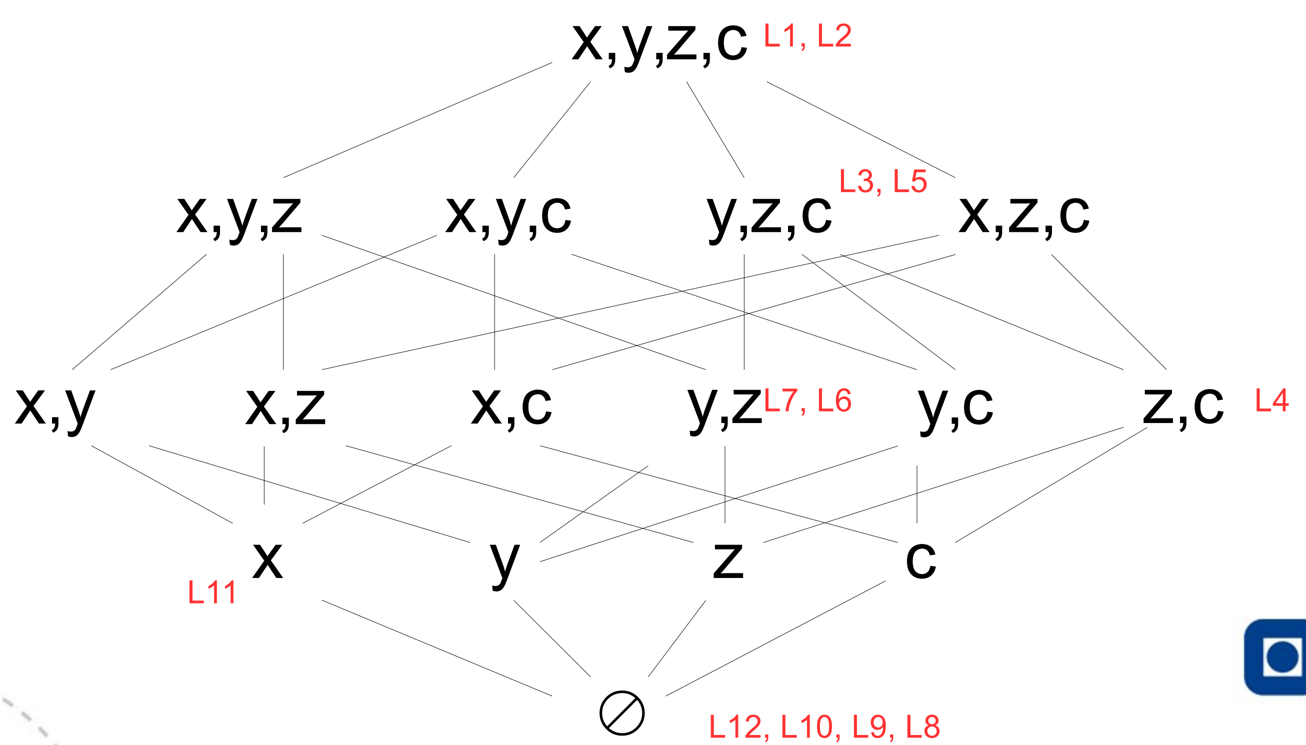
- We started them out at the bottom:



NTNU – Trondheim
Norwegian University of
Science and Technology

The program points have moved

- Now some of them are scattered around



Iteration 2

$L1 = L2 \cup \{c\}$

$L2 = L3 \cup L11$

$L3 = \{L4 - x\} \cup \{y\}$

$L4 = \{L5 - y\} \cup \{z\}$

$L5 = L6 \cup \{c\}$

$L6 = L7 \cup L9$

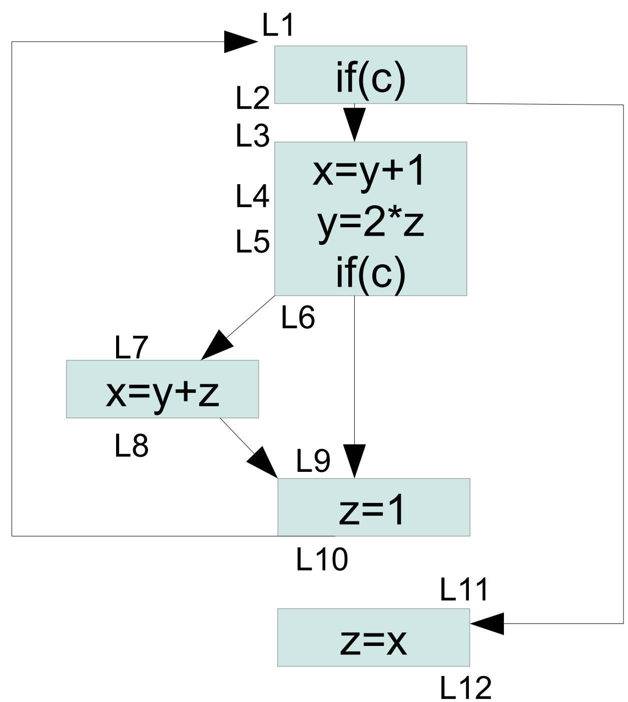
$L7 = \{L8 - x\} \cup \{y,z\}$

$L8 = L9$

$L9 = \{L10 - z\}$

$L10 = L1$

$L11 = \{L12 - z\} \cup \{x\}$



$L1 = \{x,y,z,c\}$

$L2 = \{x,y,z,c\}$

$L3 = \{y,z,c\}$

$L4 = \{\mathbf{x,}z,c\}$

$L5 = \{\mathbf{x,}y,z,c\}$

$L6 = \{\mathbf{x,}y,z\mathbf{,c\}}$

$L7 = \{y,z,\mathbf{c}\}$

$L8 = \mathbf{\{x,y,c\}}$

$L9 = \mathbf{\{x,y,c\}}$

$L10 = \mathbf{\{x,y,z,c\}}$

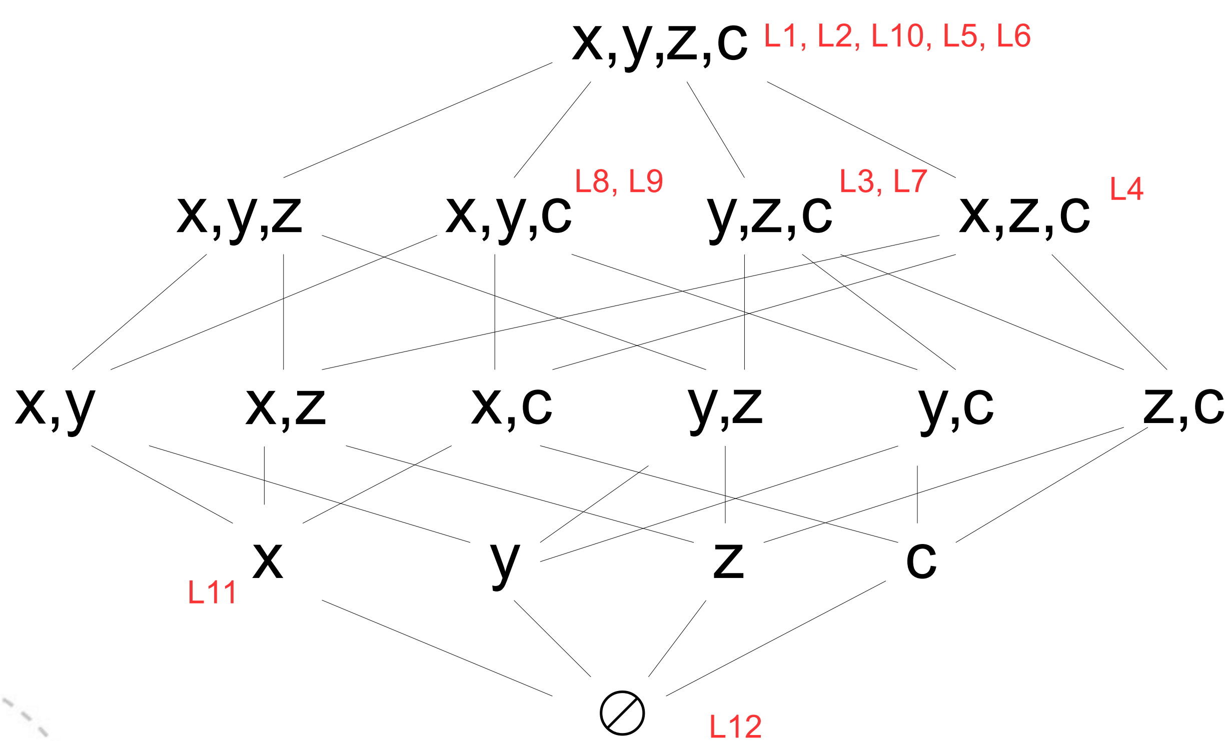
$L11 = \{x\}$

$L12 = \{\}$

NTNU – Trondheim
Norwegian University of
Science and Technology

The program points have moved again

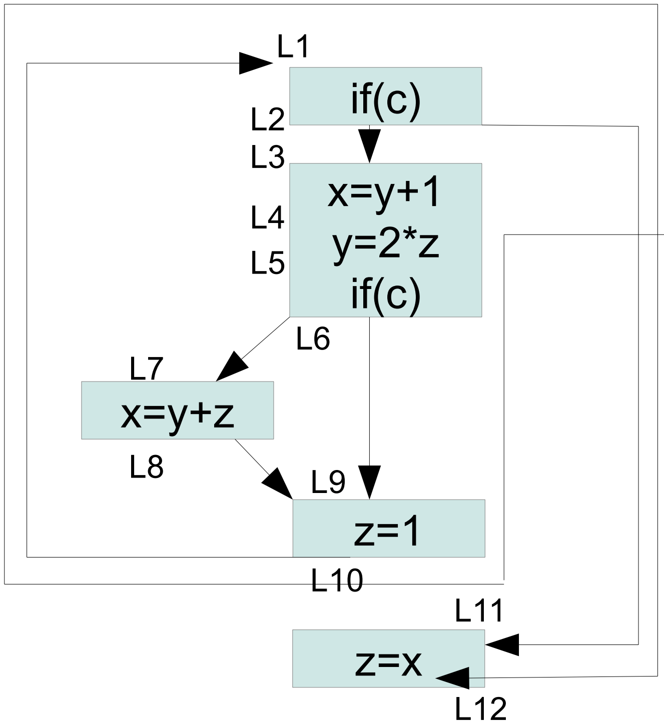
- Notice that they're only heading towards the top



NTNU – Trondheim
Norwegian University of Science and Technology

We’ve reached a fixed point

Analysis detected that there is an execution where x=y+1 is used



L1 = {x,y,z,c}
L2 = {x,y,z,c}
L3 = {y,z,c}
L4 = {**x,**z,c}
L5 = {x,y,z,c}
L6 = {x,y,z,c}
L7 = {y,z,c}
L8 = {x,y,c}
L9 = {x,y,c}
L10 = {x,y,z,c}
L11 = {x}
L12 = {}

So, the argument goes

- If the transfer function only moves program points up the lattice, they will either
  - Come to a fixed point before they reach the top of the lattice
  - Reach the top of the lattice, and have no place left to go
- Analyses that use transfer functions which have this monotonicity will always terminate at a fixed point

That was a lot of notation for a simple observation

- The goal is generality
- If liveness were all we cared about, this would be overkill
- *Reaching Definitions, Available Expressions* and *Constant Folding* are the same way, just with other choices of operators, sets, transfer functions and directions
- It's hopefully a little easier to remember them as 4 cases of 1 method rather than 4 separate approaches to separate problems

Next time

- With most of the notation in place, we'll discuss the other analysis instances within this same terminology, to highlight what they have in common, and how they differ