Compiler Construction

Lecture 16

Data-Flow Analysis

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

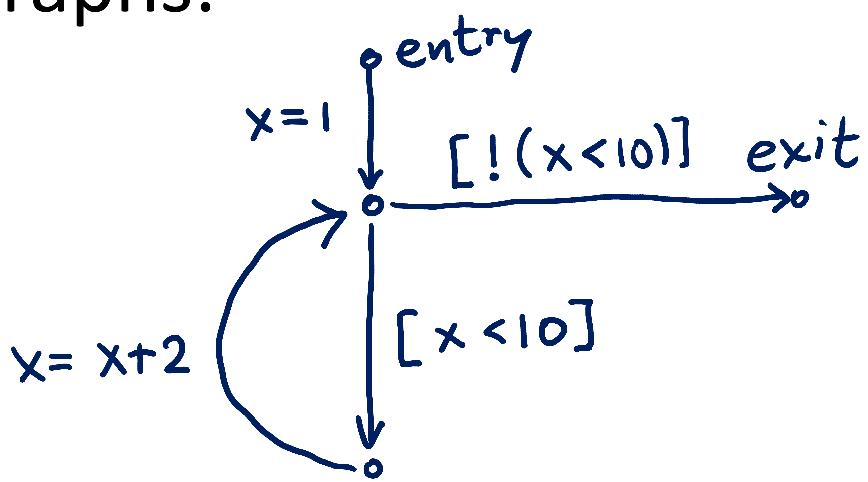
Goal of Data-Flow Analysis

Automatically compute information about the program

- Use it to report errors to user (like type errors)
- Use it to optimize the program

Works on control-flow graphs:

```
x = 1
while (x < 10) {
  x = x + 2
}
```



program

it's CFG

How We Define It

- Abstract Domain **D** (Data-Flow Facts): which information to compute?
  - Example: interval for each variable   x:[a,b], y:[a’,b’]
- Transfer Functions [[**st**]] for each statement **st**, how this statement affects the facts
  - Example:

$$[\![x = x+2]\!](x:[a,b], \dots) = (x:[a+2, b+2], \dots)$$

$x:[a,b]\quad y:[c,d]$

$x = x+2$

$x:[a+2, b+2], y:[c,d]$

Find Transfer Function: Plus

Suppose we have only two integer variables: x,y

$x: [a,b] \quad y: [c,d]$

$x = x+y$

$x: [a',b'] \quad y: [c',d']$

If $a \leq x \leq b \qquad c \leq y \leq d$

and we execute **x= x+y**

then $x' = x+y$

$y' = y$

so

$\leq x' \leq$

$\leq y' \leq$

So we can let

$a' = a+c \qquad b' = b+d$

$c' = c \qquad d' = d$

Find Transfer Function: Minus

Suppose we have only two integer variables: x,y

$x : [a, b] \quad y : [c, d]$

$y = x - y$

$x : [a', b'] \quad y : [c', d']$

If

and we execute **y= x-y**

then

So we can let

a’= a   b’ = b

c’= a - d   d’ = b - c

Transfer Functions for Tests

x : [-10, 10]

if (x > 1) {

  x :

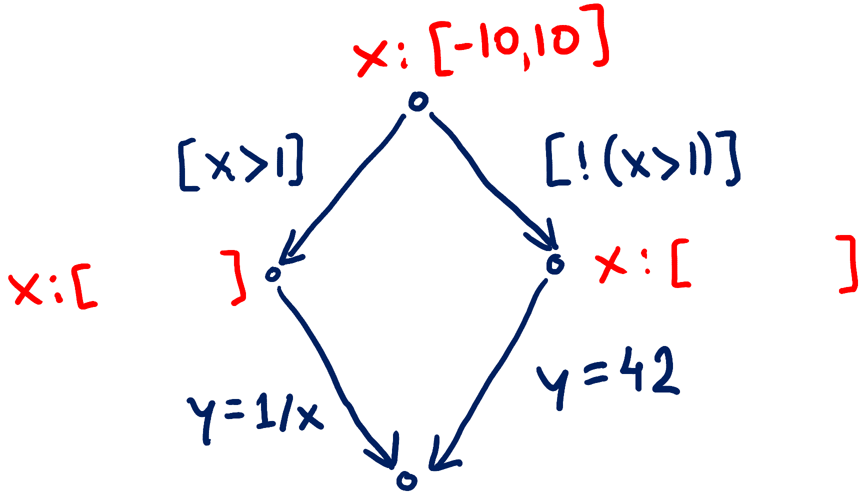
  y = 1 / x

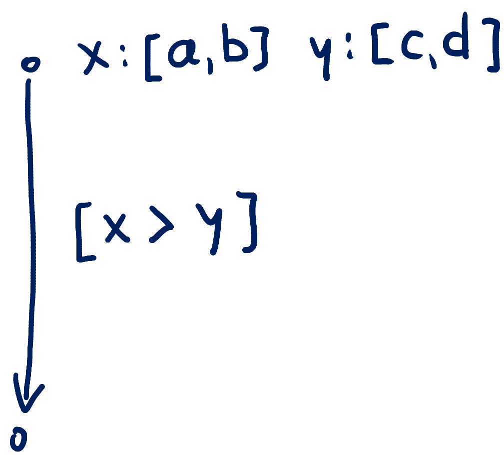
} else {

  x :

  y = 42

}





Merging Data-Flow Facts

x: [-10,10]   y: [-1000,1000]

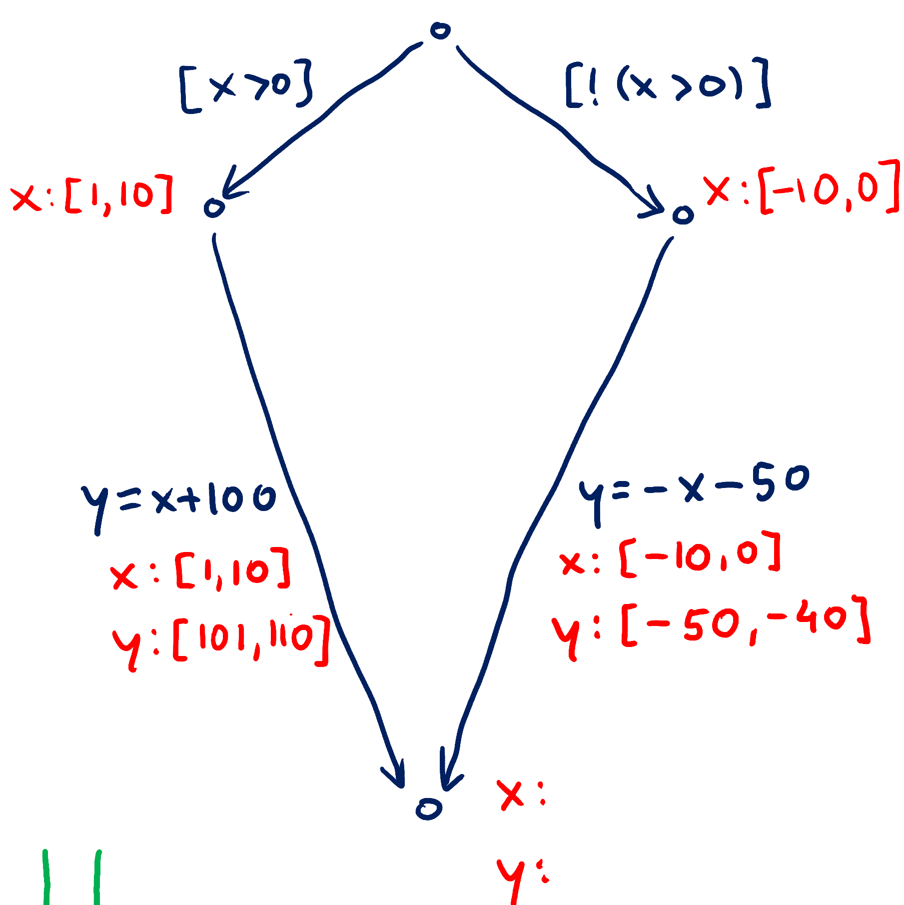
```
if (x > 0) {
```
x: y:

```
  y = x + 100
```
x: y:

```
} else {
```
x: y:

```
  y = -x – 50
```
x: y:

```
}
```
x: y:



join $\sqcup$

$[a,b] \sqcup [c,d] = [\min(a,c), \max(b,d)]$

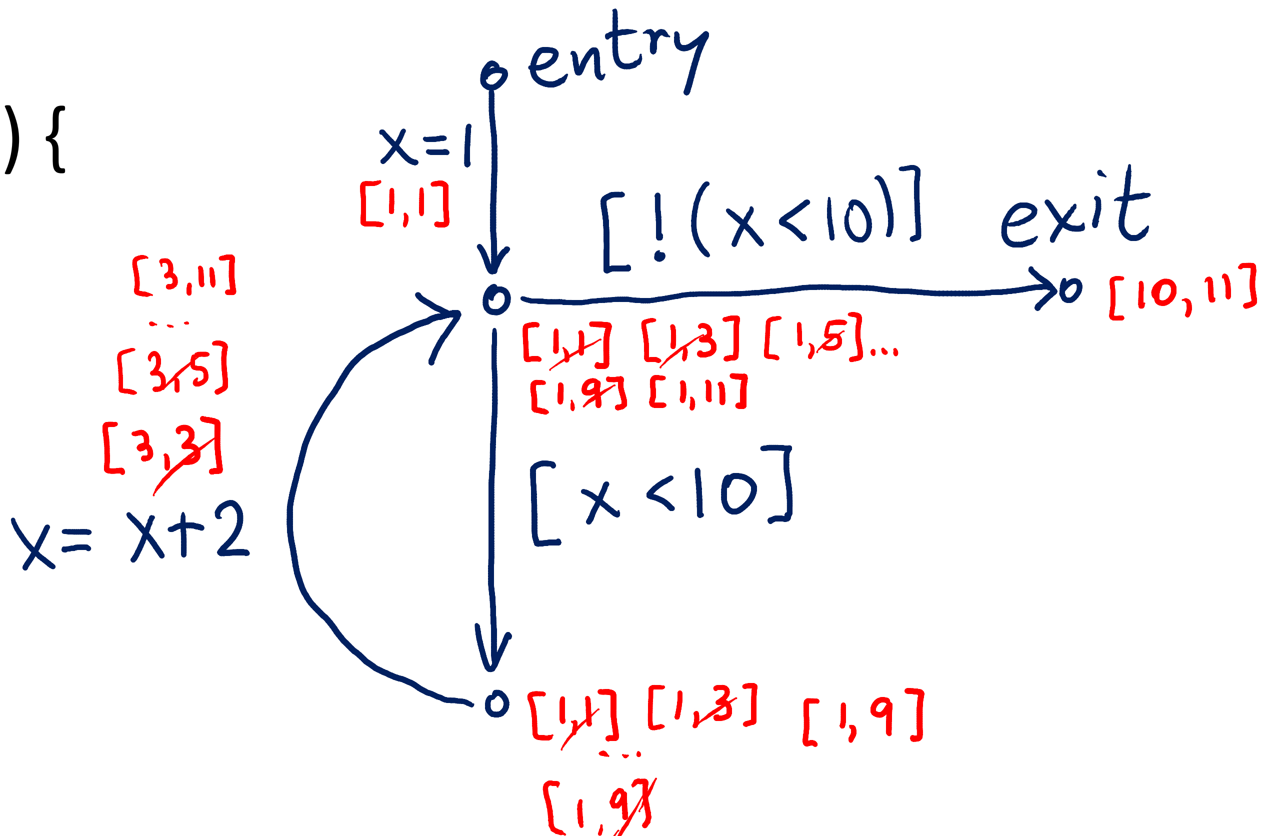
Handling Loops: Iterate Until Stabilizes

Compiler learned some facts! ☺

$[1,1] \sqcup [3,3] = [1,3]$

$[1,1] \sqcup [3,5] = [1,5]$

```
x = 1
```
$x \in [1,1]$
```
while (x < 10) {
```
$x \in [1,9]$
```
  x = x + 2
```
$x \in [3,11]$
```
}
```
$x \in [10,11]$



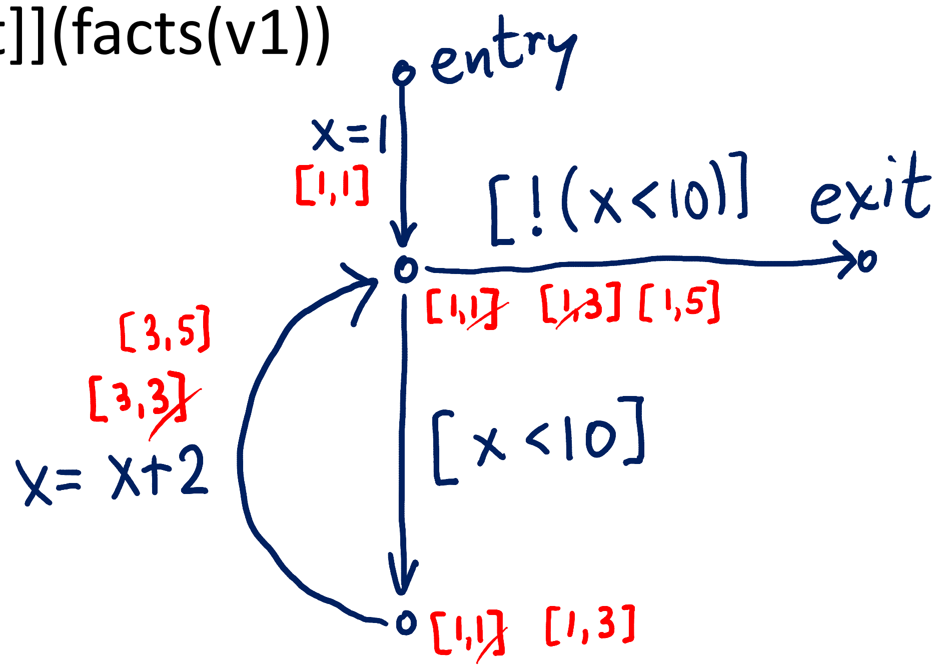
Data-Flow Analysis Algorithm

```
var facts : Map[Vertex,Domain] = Map.withDefault(empty)
facts(entry) = initialValues // change
while (there was change)
  pick edge (v1,statmt,v2) from CFG
         such that facts(v1) was changed
  facts(v2)=facts(v2) join [[statmt]](facts(v1))
}
```

$[1,1] \sqcup [3,3] = [1,3]$

$[1,1] \sqcup [3,5] = [1,5]$

Order does not matter for the end result, as long as we do not permanently neglect any edge whose source was changed.



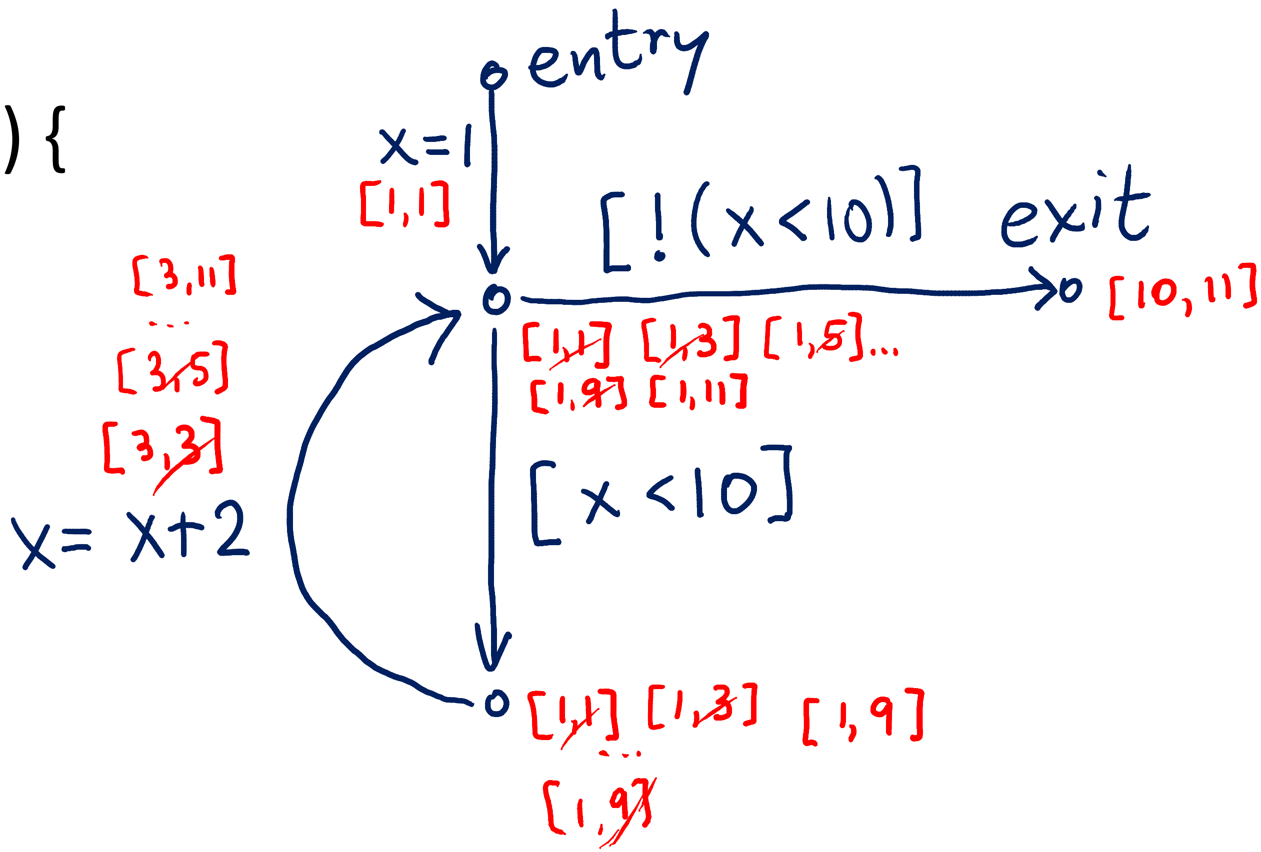
Handling Loops: Iterate Until Stabilizes

Compiler learned some facts! ☺

$[1,1] \sqcup [3,3] = [1,3]$

$[1,1] \sqcup [3,5] = [1,5]$

```
x = 1
```
$x \in [1,1]$
```
while (x < 10) {
```
$x \in [1,9]$
```
  x = x + 2
```
$x \in [3,11]$
```
}
```
$x \in [10,11]$



Handling Loops: Iterate **Until Stabilizes**

Compiler learns
some facts, but only after long time

```
x = 1
n = 100000
while (x < n) {
  x = x + 2
}
```

Handling Loops: Iterate **Until Stabilizes**

For unknown program inputs it may be practically impossible to know how long it takes

```
var x : BigInt = 1
var n : BigInt = readInput()
while (x < n) {
  x = x + 2
}
```

Solutions

- smaller domain, e.g. only certain intervals [a,b] where a,b in {-∞,-127,-1,0,1,127,∞}
- *widening* techniques (make it less precise on demand)

Size of analysis domain

Interval analysis:

$D_1 = \{ [a,b] \mid a \leq b,\ a,b \in \{-M,-127,-1,0,1,127,M-1\}\} \cup \{\perp\}$

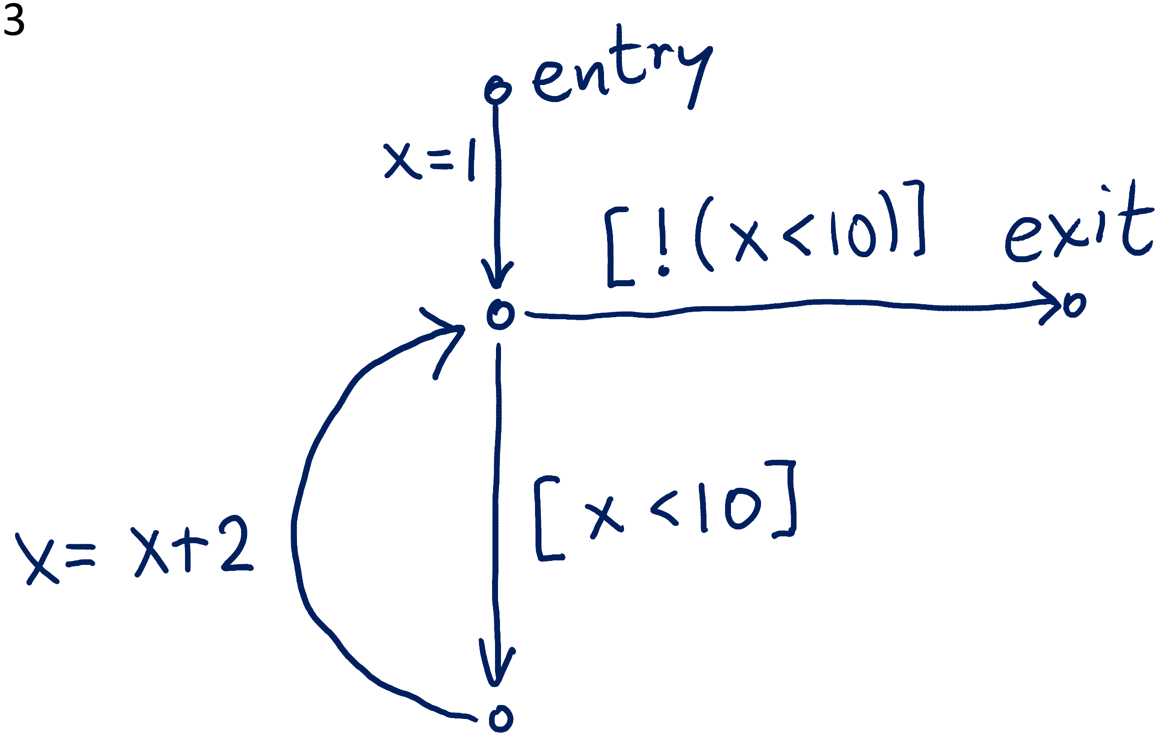
Constant propagation:

$D_2 = \{ [a,a] \mid a \in \{-M,-(M-1),\ldots,-2,-1,0,1,2,3,\ldots,M-1\}\} \cup \{\perp\}$

suppose M is 2^{63}

$|D_1| =$

$|D_2| =$



How many steps does the analysis take to finish (converge)?

Interval analysis:

$D_1 = \{ [a,b] \mid a \leq b,\ a,b \in \{-M,-127,-1,0,1,127,M-1\}\} \cup \{\perp\}$

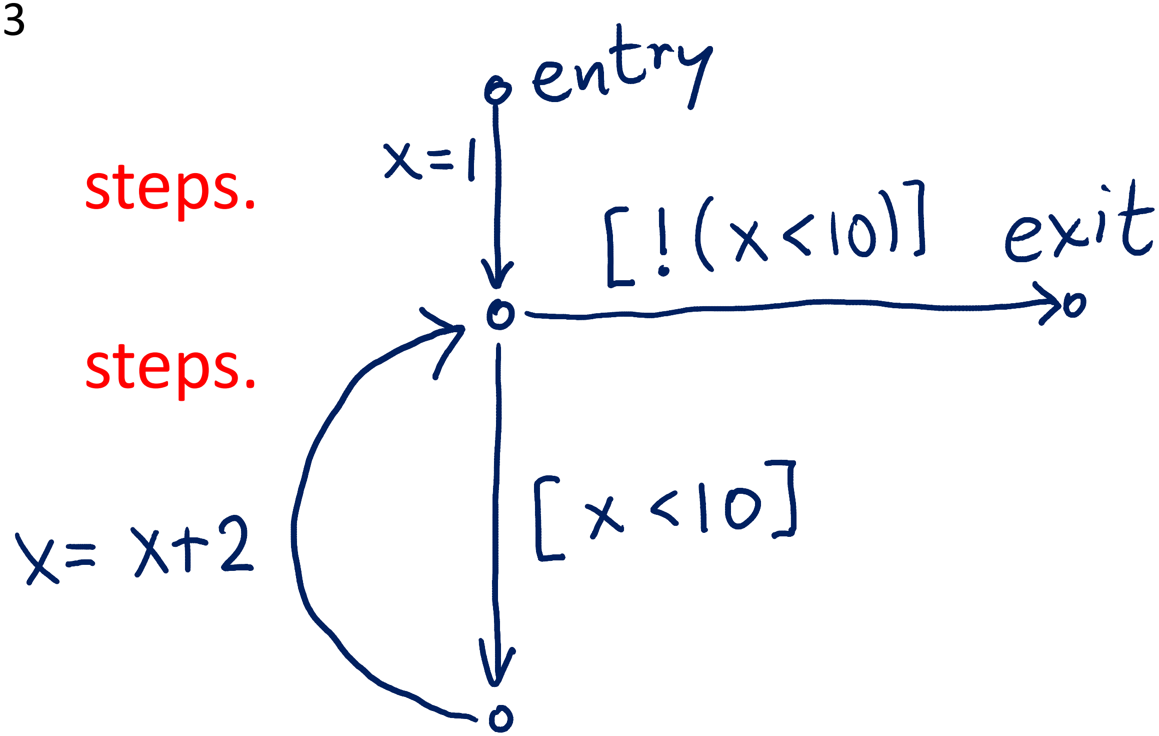
Constant propagation:

$D_2 = \{ [a,a] \mid a \in \{-M,-(M-1),\ldots,-2,-1,0,1,2,3,\ldots,M-1\}\} \cup \{\perp\}$

suppose M is 2^{63}

With D_1 takes at most steps.

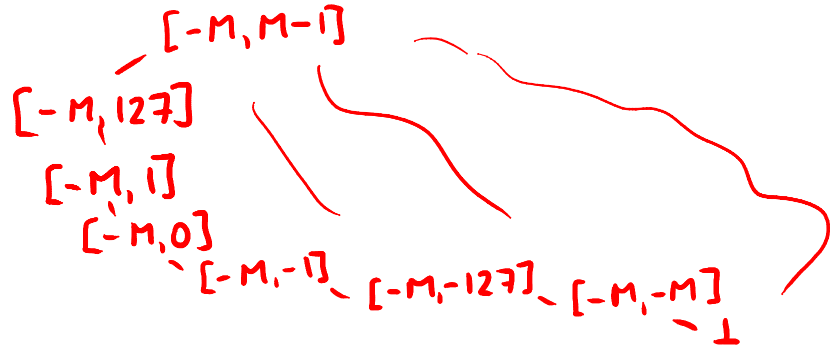
With D_2 takes at most steps.



Termination Given by Length of Chains

Interval analysis:

$D_1 = \{ [a,b] \mid a \leq b,\ a,b \in \{-M,-127,-1,0,1,127,M-1\}\} \cup \{\perp\}$

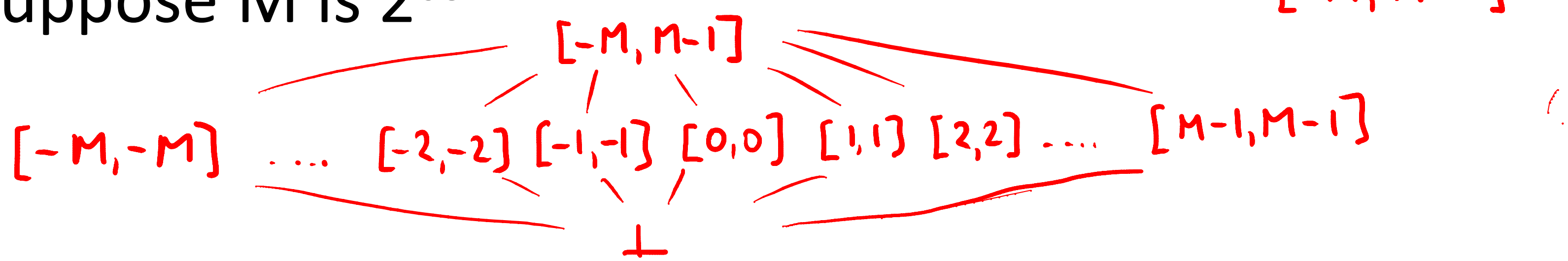


Constant propagation:

$D_2 = \{ [a,a] \mid a \in \{-M,\ldots,-2,-1,0,1,2,3,\ldots,M-1\}\} \cup \{\perp\} \cup \{T\}$

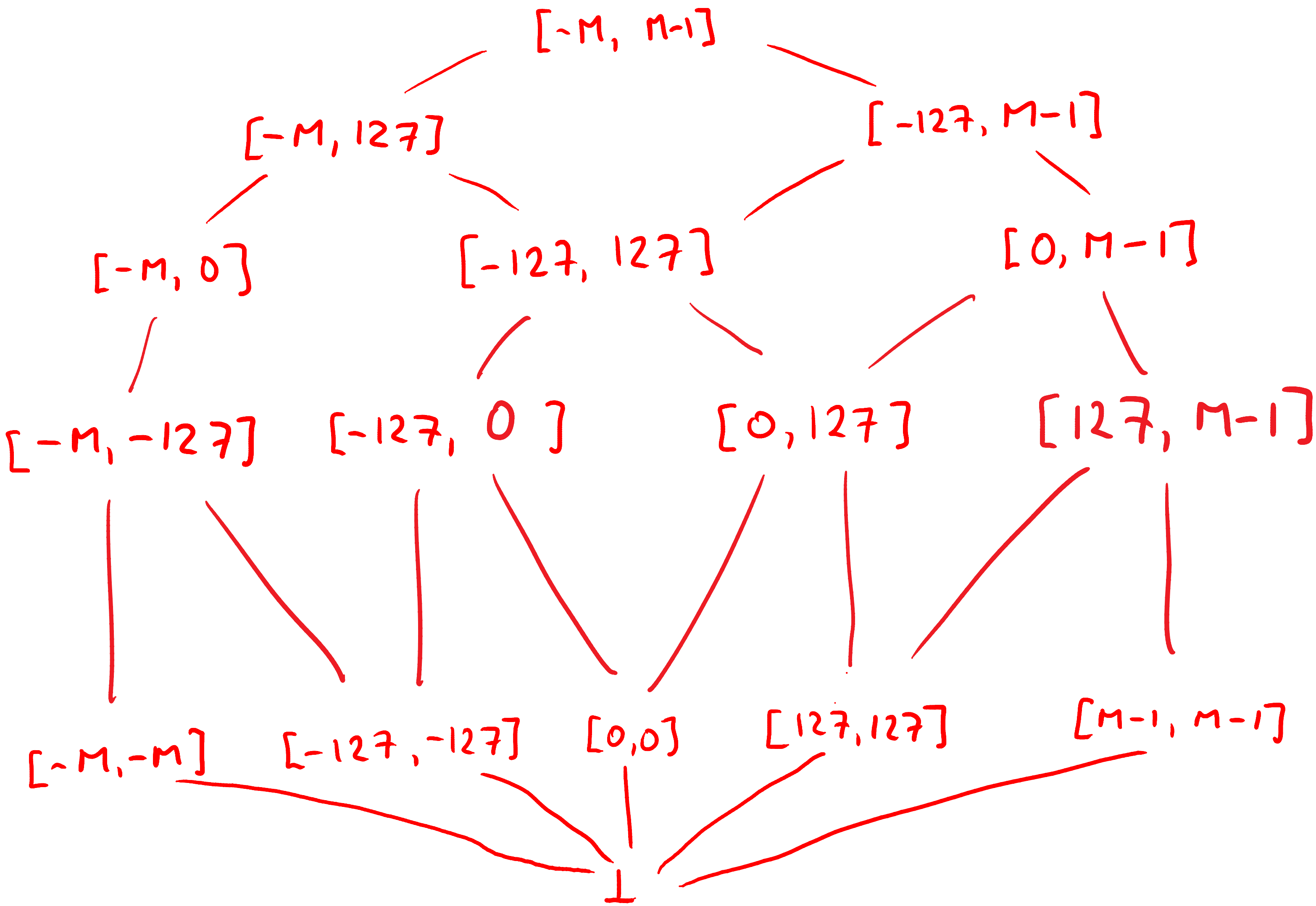
suppose M is 2^{63}

[-M, M-1]



Domain is a **lattice**. Maximal chain length = **lattice height**

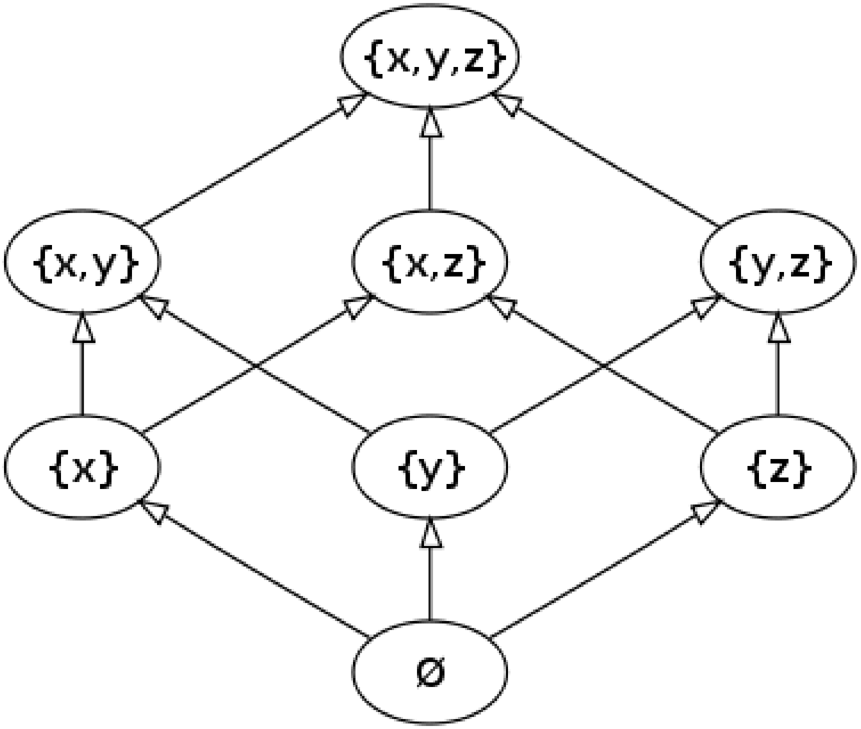
Lattice for intervals [a,b] where a,b∈{-M,-127,0,127,M-1}



Lattice

Partially ordered set (D, $\leq$)

- Every $a, b \in D$ there exists the least element c s.t. a $\leq c$, b $\leq c$ (lub, join, $\sqcup$)
- It has a top (T) element and a bottom element ($\perp$)



Lattice for ($\wp(\{x, y, z\})$, $\subseteq$)

Data-Flow Analysis Ingredients

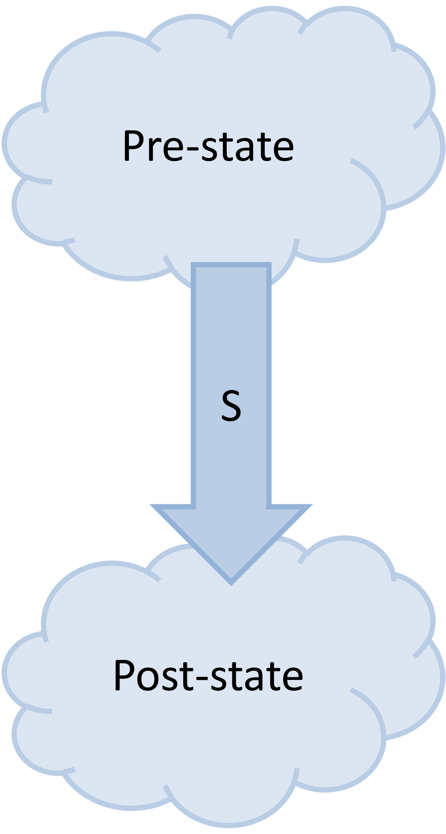
Given some concrete domain D_C:

- Abstract Domain D_A forming a lattice
  - An Abstraction Function $D_C \longmapsto D_A$
  - A Concretization Function $D_A \longmapsto D_C$
- The program semantics within D_A:

  A transfer function [[_]] : $Stmts \longmapsto (D_A \longmapsto D_A)$

Transfer Function

Given a statement S and an abstract pre-state, compute the abstract post-state.



Needs to be monotonous:

$$A_1 \sqsubseteq A_2 \;\Rightarrow [[S]](A_1) \sqsubseteq [[S]](A_2)$$

Abstraction/Concretization