IC637 Program Analysis

Lecture 4: Octagon Domain

Minseok Jeon

2025 Fall

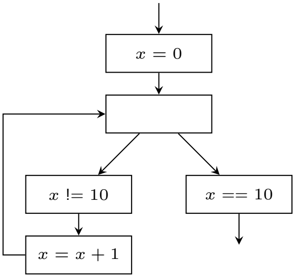
Review: Interval Domain

- **Interval Domain**: Abstract domain $\hat{\mathbb{Z}} = \{\bot\} \cup \{[l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, \infty\}, l \leq u\}$
  - Partial order: $\bot \sqsubseteq x$, $[l_1, u_1] \sqsubseteq [l_2, u_2]$ iff $l_2 \leq l_1$ and $u_1 \leq u_2$
  - Join: $[l_1, u_1] \sqcup [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$
  - Meet: $[l_1, u_1] \sqcap [l_2, u_2] = [\max(l_1, l_2), \min(u_1, u_2)]$ if overlap, $\bot$ otherwise
- **Worklist Algorithm**: Iterative fixed-point computation
  - Widening phase: Apply widening at loop headers until convergence
  - Narrowing phase: Apply narrowing to refine results
- **Widening & Narrowing**: Essential for termination in infinite domains
  - Widening (∇): Extrapolates to infinity to ensure convergence
  - Narrowing ($\triangle$): Refines over-approximations from widening
  - Widening with thresholds: Uses predefined values to improve precision
- **Key Insight**: Balance between precision and scalability (e.g., termination)

Example

- Describe the result of the interval analysis:
  1. without widening
  2. with widening/narrowing

```
void main(){
    int x = 0;
    while (x != 10) {
        x = x + 1;
    }
}
```

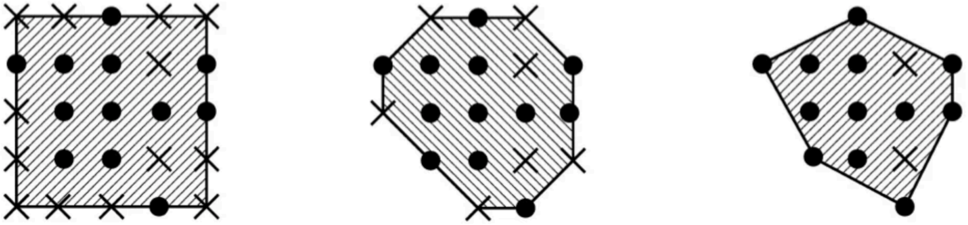


Discussion

- Give an example program that cannot be precisely analyzed by interval domain.

Relational Abstract Domains

- Intervals vs Octacons vs Polyhedra



The Octagon Domain

- Focus: Core idea of the Octagon domain

```
void main(){
    int a[10];
    x = 0; y = 0;
    while (x < 9) {
        x++; y++;
    }
    a[y] = 0;
}
```

Octagon analysis :

$x : [9,9]$
$y : [9,9]$
$x - y : [0,0]$
$x + y : [18,18]$

Interval analysis :

$x : [9,9]$
$y : [0,\infty]$

Difference Bound Matrix (DBM)

- $(N+1) \times (N+1)$ matrix (N: number of variables) e.g.,

$$\begin{array}{c|ccc} & 0 & x & y \\ \hline 0 & 0-0 & x-0 & y-0 \\ x & 0-x & x-x & y-x \\ y & 0-y & x-y & y-y \end{array}$$

- Example

$$\begin{array}{c|ccc} & 0 & x & y \\ \hline 0 & 0 & 10 & 10 \\ x & 0 & 0 & 0 \\ y & 0 & 0 & 0 \end{array} \iff \begin{array}{r} 0 \le x \le 10 \\ 0 \le y \le 10 \\ y - x \le 0 \\ x - y \le 0 \end{array} \qquad \begin{array}{c|ccc} & 0 & x & y \\ \hline 0 & 0 & 10 & \infty \\ x & -1 & 0 & -1 \\ y & 0 & 1 & 0 \end{array} \iff \begin{array}{r} 1 \le x \le 10 \\ 0 \le y \\ y - x \le -1 \\ x - y \le 1 \end{array}$$

Difference Bound Matrix (DBM)

- A DBM represents a set of program states (N-dim points)

$$\gamma\left(\begin{bmatrix} 0 & 10 & \infty \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}\right) = \{(x, y) \mid 1 \leq x \leq 10,\ 0 \leq y,\ y - x \leq -1,\ x - y \leq 1\}$$

$$\gamma\left(\begin{bmatrix} 0 & 5 & \infty \\ \infty & 0 & -1 \\ 0 & 2 & 0 \end{bmatrix}\right) =$$

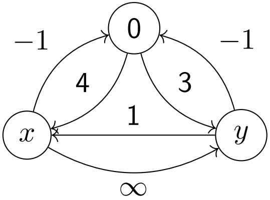
- Quiz: find a matrix M such that $\gamma(M) = \emptyset$
- Question: can two different DBMs represent the same set of points?

Difference Bound Matrix (DBM)

- A DBM can also be represented by a directed graph

	0	x	y
0	0	4	3
x	-1	0	$+\infty$
y	-1	1	0

$\iff$



	0	x	y
0	0	10	∞
x	-1	0	-1
y	0	1	0

$\iff$

Difference Bound Matrix (DBM)

- Two different DBMs can represent the same set of points

$$\gamma \begin{pmatrix} +\infty & 4 & 3 \\ -1 & +\infty & +\infty \\ -1 & 1 & +\infty \end{pmatrix} = \gamma \begin{pmatrix} 0 & 5 & 3 \\ -1 & +\infty & +\infty \\ -1 & 1 & +\infty \end{pmatrix}$$

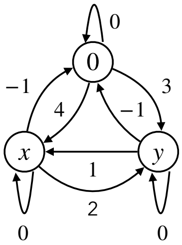
- How can we check if two DBMs represent the same set of points?

Difference Bound Matrix (DBM)

- Closure (normalization) via the Floyd-Warshall algorithm

$$\begin{bmatrix} +\infty & 4 & 3 \\ -1 & +\infty & +\infty \\ -1 & 1 & +\infty \end{bmatrix}^* = \begin{bmatrix} 0 & 4 & 3 \\ -1 & 0 & 2 \\ -1 & 1 & 0 \end{bmatrix}$$

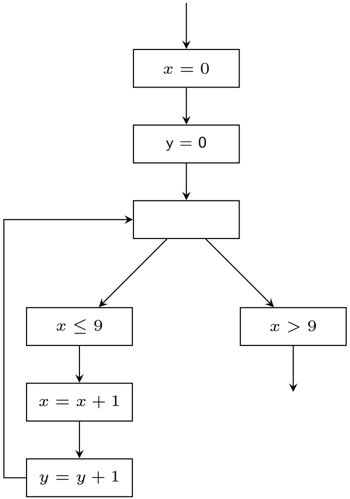
$$\begin{bmatrix} 0 & 5 & 3 \\ -1 & +\infty & +\infty \\ -1 & 1 & +\infty \end{bmatrix}^* = \begin{bmatrix} 0 & 4 & 3 \\ -1 & 0 & 2 \\ -1 & 1 & 0 \end{bmatrix}$$



for $n = 0$ **to** $n-1$ **do**
  $dist[n][n] = 0$
for $k = 0$ **to** $n-1$ **do**
  for $i = 0$ **to** $n-1$ **do**
    for $j = 0$ **to** $n-1$ **do**
      $dist[i][j] \leftarrow \min(dist[i][j], dist[i][k] + dist[k][j])$

Fixed Point Computation with Widening

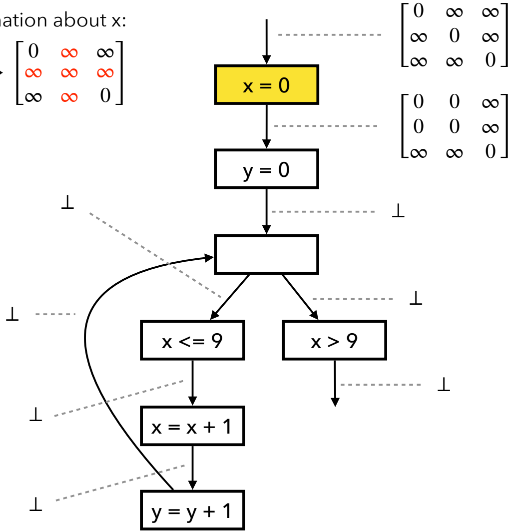
```
int x = 0;
int y = 0;
while (x <= 9) {
    x = x + 1;
    y = y + 1;
}
```



Fixed Point Computation with Widening/Narrowing

1. Remove information about x:

$$\begin{bmatrix} 0 & \infty & \infty \\ \infty & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & \infty & \infty \\ \infty & \infty & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

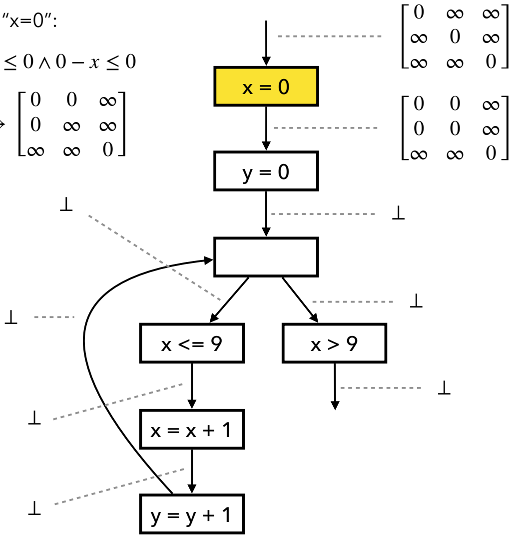


Fixed Point Computation with Widening/Narrowing

2. Add constraint "x=0":

$$x = 0 \iff x - 0 \leq 0 \wedge 0 - x \leq 0$$

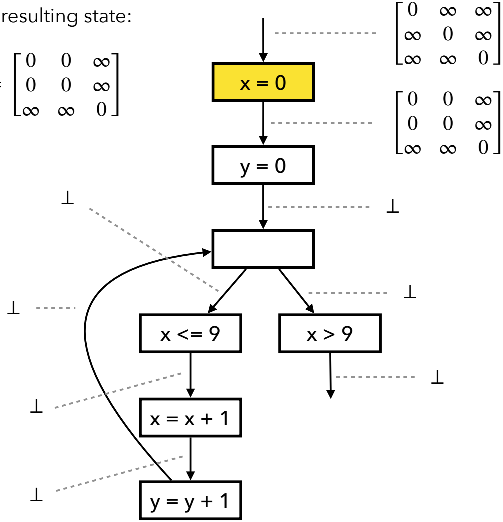
$$\begin{bmatrix} 0 & \infty & \infty \\ \infty & \infty & \infty \\ \infty & \infty & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & \infty \\ 0 & \infty & \infty \\ \infty & \infty & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

3. Normalize the resulting state:

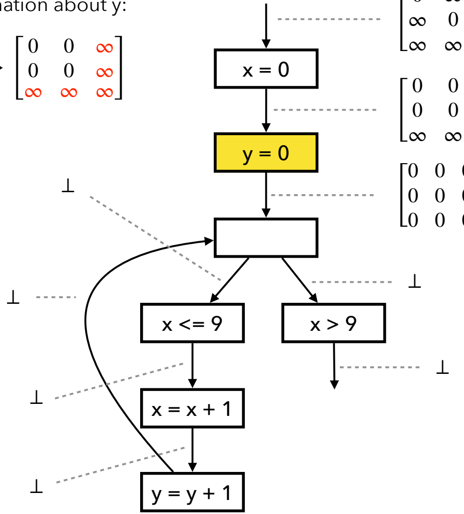
$$\begin{bmatrix} 0 & 0 & \infty \\ 0 & \infty & \infty \\ \infty & \infty & 0 \end{bmatrix}^* = \begin{bmatrix} 0 & 0 & \infty \\ 0 & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

1. Remove information about y:

$$\begin{bmatrix} 0 & 0 & \infty \\ 0 & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & \infty \\ 0 & 0 & \infty \\ \infty & \infty & \infty \end{bmatrix}$$



$$\begin{bmatrix} 0 & \infty & \infty \\ \infty & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

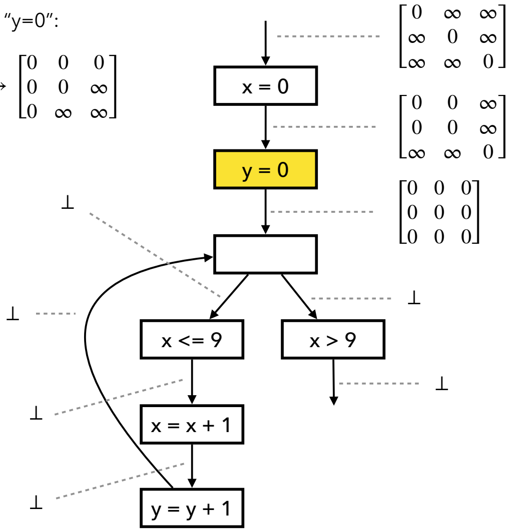
$$\begin{bmatrix} 0 & 0 & \infty \\ 0 & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Fixed Point Computation with Widening/Narrowing

2. Add constraint "y=0":

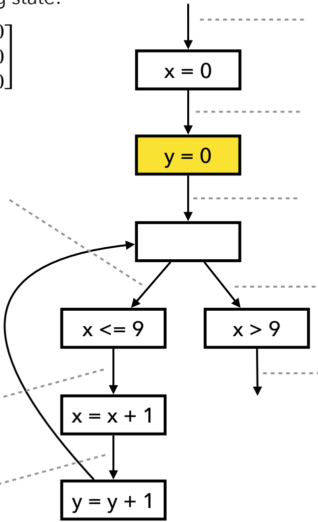
$$\begin{bmatrix} 0 & 0 & \infty \\ 0 & 0 & \infty \\ \infty & \infty & \infty \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \infty \\ 0 & \infty & \infty \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

3. Normalize the resulting state:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \infty \\ 0 & \infty & \infty \end{bmatrix}^* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



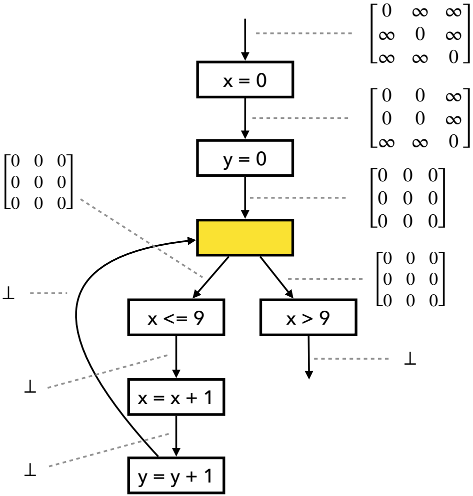
$$\begin{bmatrix} 0 & \infty & \infty \\ \infty & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & \infty \\ 0 & 0 & \infty \\ \infty & \infty & 0 \end{bmatrix}$$

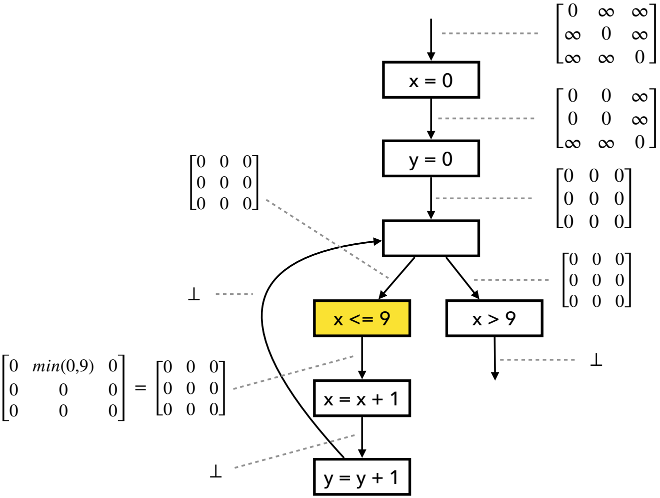
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$\perp$ $\perp$ $\perp$ $\perp$ $\perp$ $\perp$

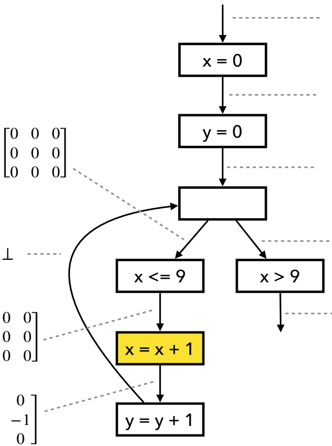
Fixed Point Computation with Widening/Narrowing



Fixed Point Computation with Widening/Narrowing



Fixed Point Computation with Widening/Narrowing



$$\begin{bmatrix}0&\infty&\infty\\\infty&0&\infty\\\infty&\infty&0\end{bmatrix}$$

$$\begin{bmatrix}0&0&\infty\\0&0&\infty\\\infty&\infty&0\end{bmatrix}$$

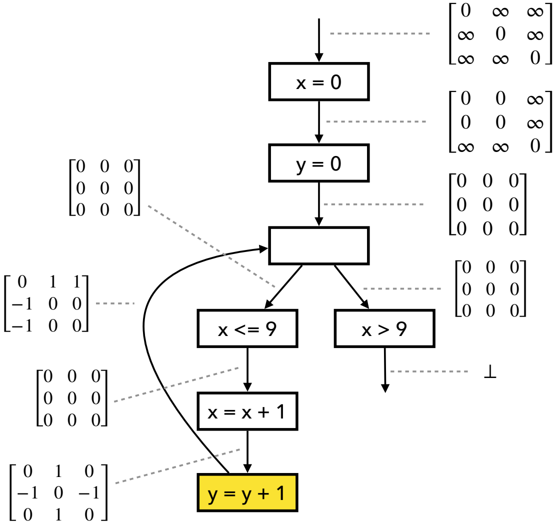
$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}$$

$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \rightarrow \begin{bmatrix}0&1&0\\-1&0&-1\\0&1&0\end{bmatrix}$$

$x - x' \leq c \rightarrow x - x' \leq c + 1$

$x' - x \leq c \rightarrow x' - x \leq c - 1$

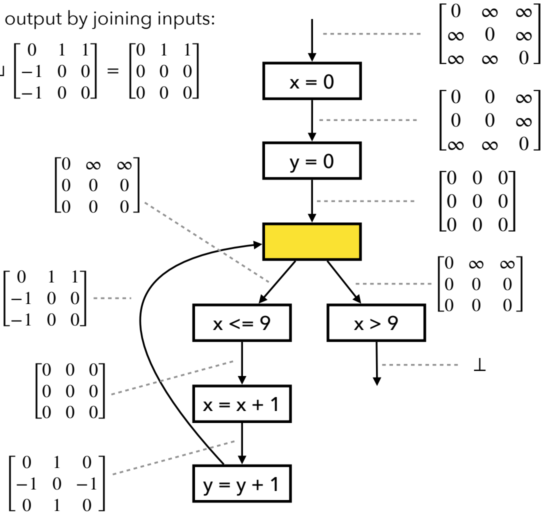
Fixed Point Computation with Widening/Narrowing



Fixed Point Computation with Widening/Narrowing

1. Compute output by joining inputs:

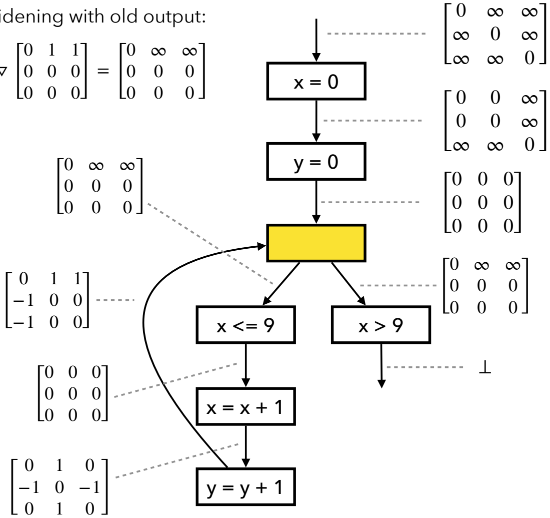
$$\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix} \sqcup \begin{bmatrix}0&1&1\\-1&0&0\\-1&0&0\end{bmatrix} = \begin{bmatrix}0&1&1\\0&0&0\\0&0&0\end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

2. Apply widening with old output:

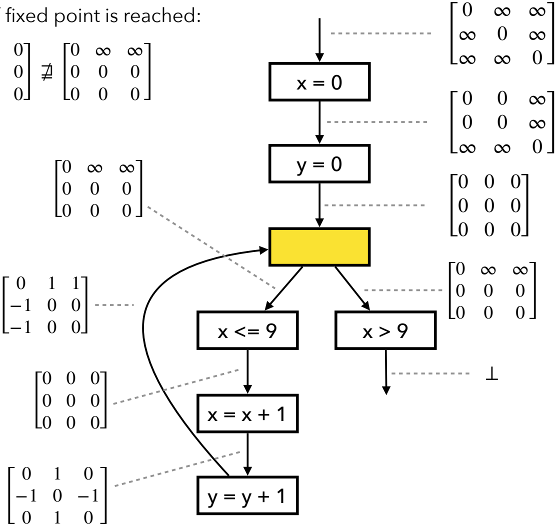
$$\begin{bmatrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix} \triangledown \begin{bmatrix}0 & 1 & 1\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & \infty & \infty\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

3. Check if fixed point is reached:

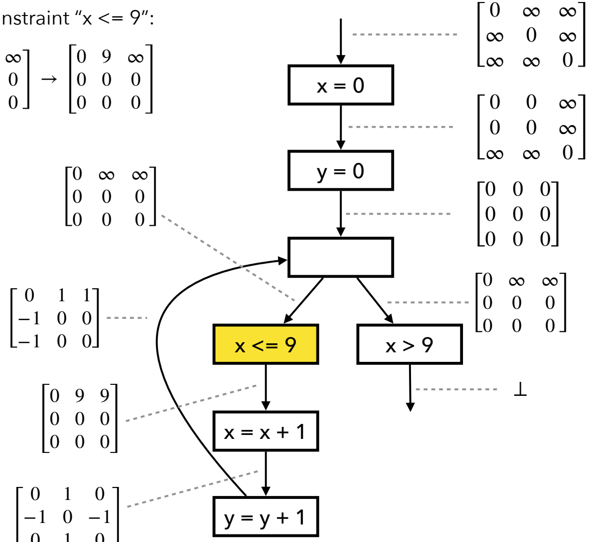
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \not\sqsupseteq \begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

1. Add constraint “x <= 9”:

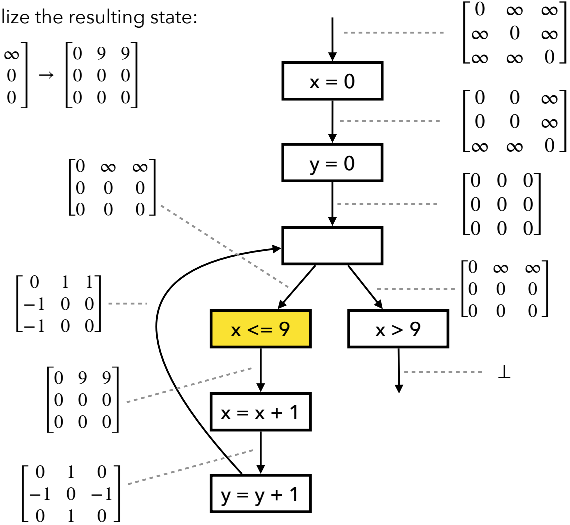
$$\begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 9 & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



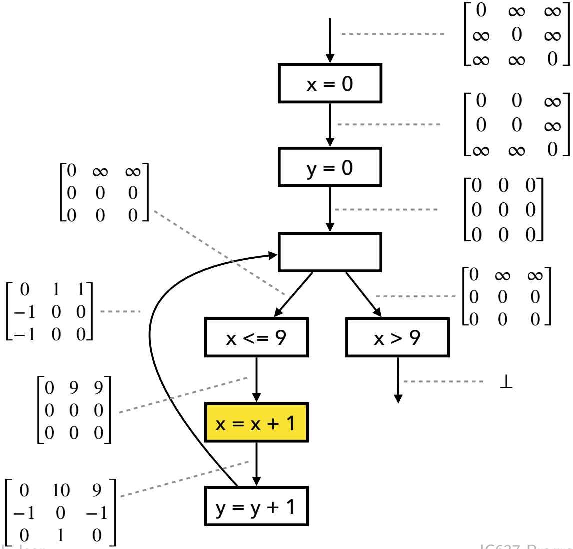
Fixed Point Computation with Widening/Narrowing

2. Normalize the resulting state:

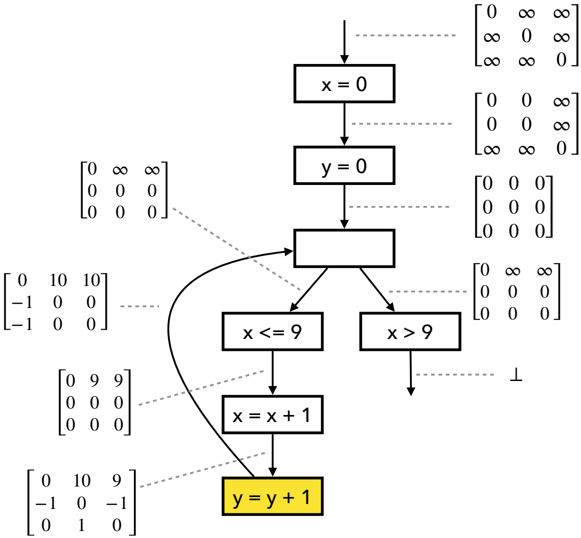
$$\begin{bmatrix} 0 & 9 & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 9 & 9 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing



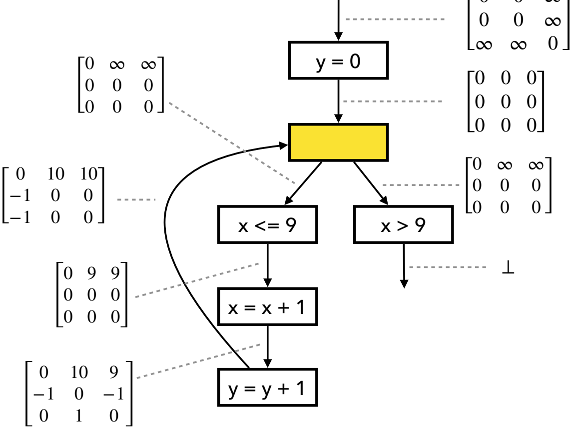
Fixed Point Computation with Widening/Narrowing



Fixed Point Computation with Widening/Narrowing

1. Compute output by joining inputs:

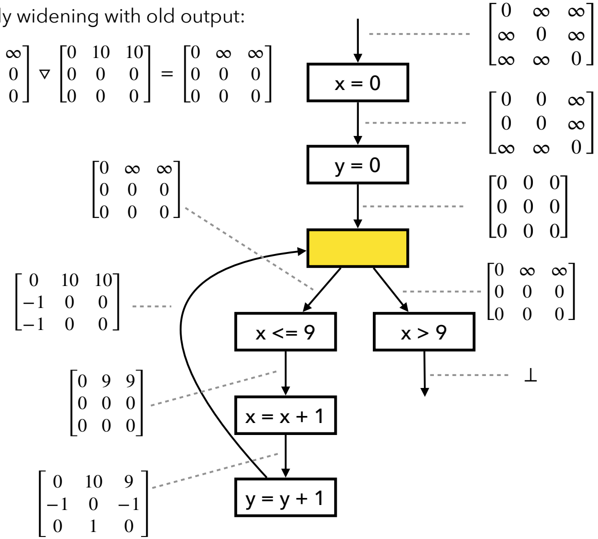
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sqcup \begin{bmatrix} 0 & 10 & 10 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 10 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

2. Apply widening with old output:

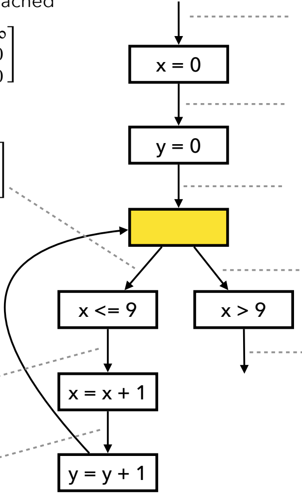
$$\begin{bmatrix}0&\infty&\infty\\0&0&0\\0&0&0\end{bmatrix} \triangledown \begin{bmatrix}0&10&10\\0&0&0\\0&0&0\end{bmatrix} = \begin{bmatrix}0&\infty&\infty\\0&0&0\\0&0&0\end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

3. Check if fixed point is reached

$$\begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sqsupseteq \begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

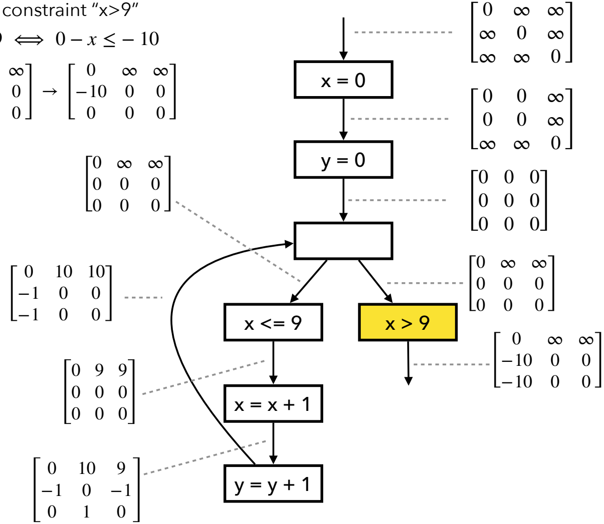


Fixed Point Computation with Widening/Narrowing

1. Add constraint "x>9"

$$x > 9 \iff 0 - x \leq -10$$

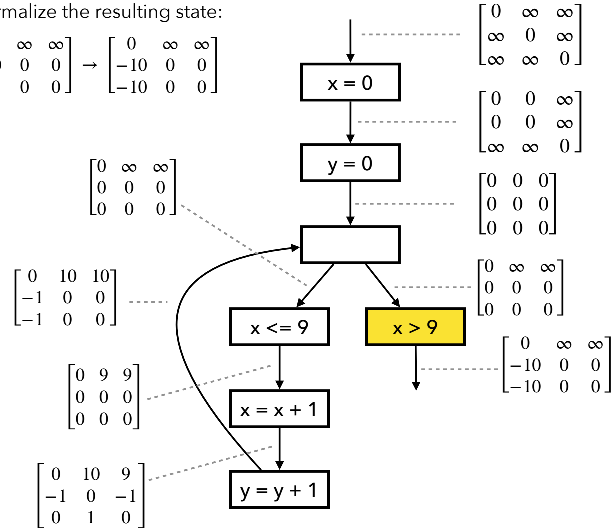
$$\begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & \infty & \infty \\ -10 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

2. Normalize the resulting state:

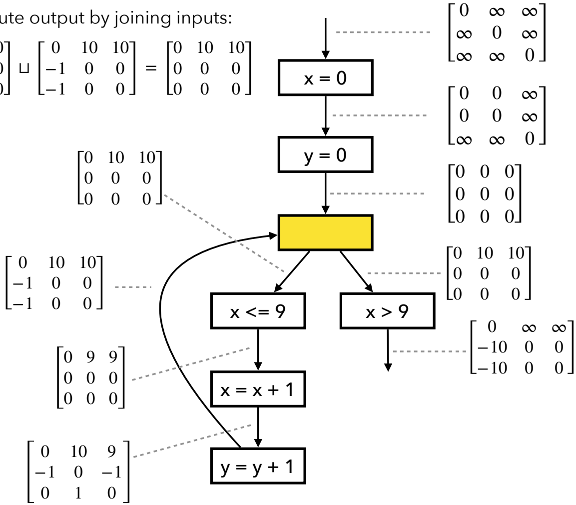
$$\begin{bmatrix} 0 & \infty & \infty \\ -10 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \to \begin{bmatrix} 0 & \infty & \infty \\ -10 & 0 & 0 \\ -10 & 0 & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

1. Compute output by joining inputs:

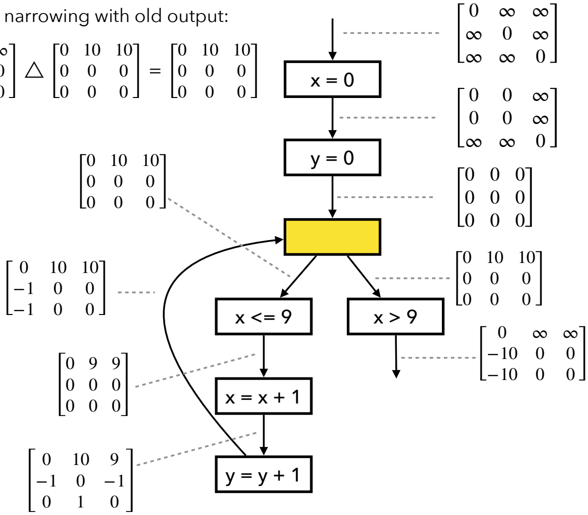
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sqcup \begin{bmatrix} 0 & 10 & 10 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 10 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Fixed Point Computation with Widening/Narrowing

2. Apply narrowing with old output:

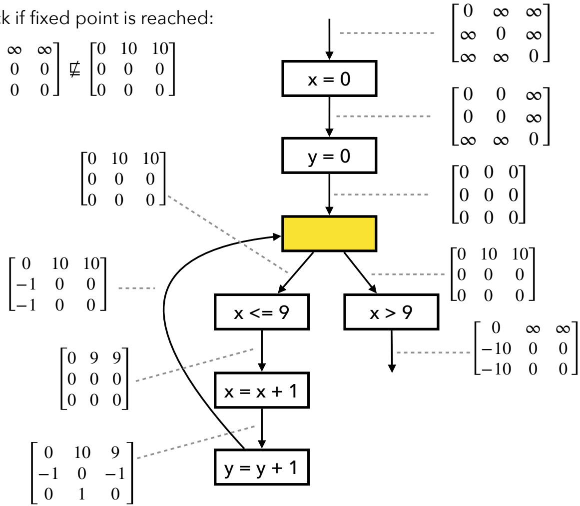
$$\begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \triangle \begin{bmatrix} 0 & 10 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 10 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



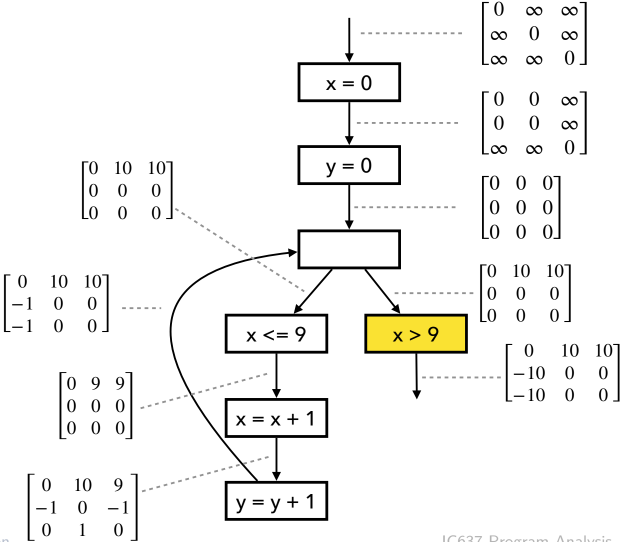
Fixed Point Computation with Widening/Narrowing

3. Check if fixed point is reached:

$$\begin{bmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \not\sqsubseteq \begin{bmatrix} 0 & 10 & 10 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

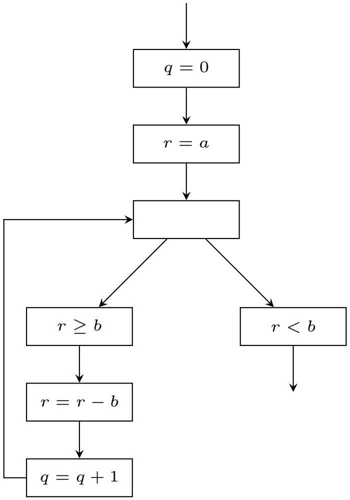


Fixed Point Computation with Widening/Narrowing



Motivating Example

```
//a >= 0; b >= 0;
int q = 0;
int r = a;
while (r >= b) {
    r = r - b;
    q = q + 1;
}
assert(q >= 0);
assert(r >= 0);
```



Static Analysis Use Cases: Infer

- `https://github.com/facebook/infer/`

Infer build failing website failing

Infer is a static analysis tool for Java, C++, Objective-C, and C. Infer is written in OCaml.

- Running Infer: e.g.,
  - infer capture – make
  - infer analyze

Infer's Intermediate Language

- https://github.com/facebook/infer/blob/main/infer/src/IR/Sil.mli

```
type instr =
  | Load of {id: Ident.t; e: Exp.t; typ: Typ.t; loc: Location.t}
      (** Load a value from the heap into an identifier.

          [id = *e:typ] where

          - [e] is an expression denoting a heap address
          - [typ] is the type of [*e] and [id]. *)
  | Store of {e1: Exp.t; typ: Typ.t; e2: Exp.t; loc: Location.t}
      (** Store the value of an expression into the heap.

          [*e1:typ = e2] where

          - [e1] is an expression denoting a heap address
          - [typ] is the type of [*e1] and [e2]. *)
```

Summary

- **Static Analysis Principles**
  - Choose appropriate abstract domains (intervals, octagons, polyhedra)
  - Balance precision vs. scalability

- **Interval Domain**
  - Simple non-relational domain: $[l, u]$ bounds
  - Widening/narrowing for fixed-point computation
  - Limited expressiveness for relational properties

- **Octagon Domain**
  - Relational domain using Difference Bound Matrices (DBM)
  - Constraints: $\pm x_i \pm x_j \leq c$
  - Floyd-Warshall algorithm for closure/normalization

- **Practical Tools**
  - Facebook Infer: Industrial static analyzer