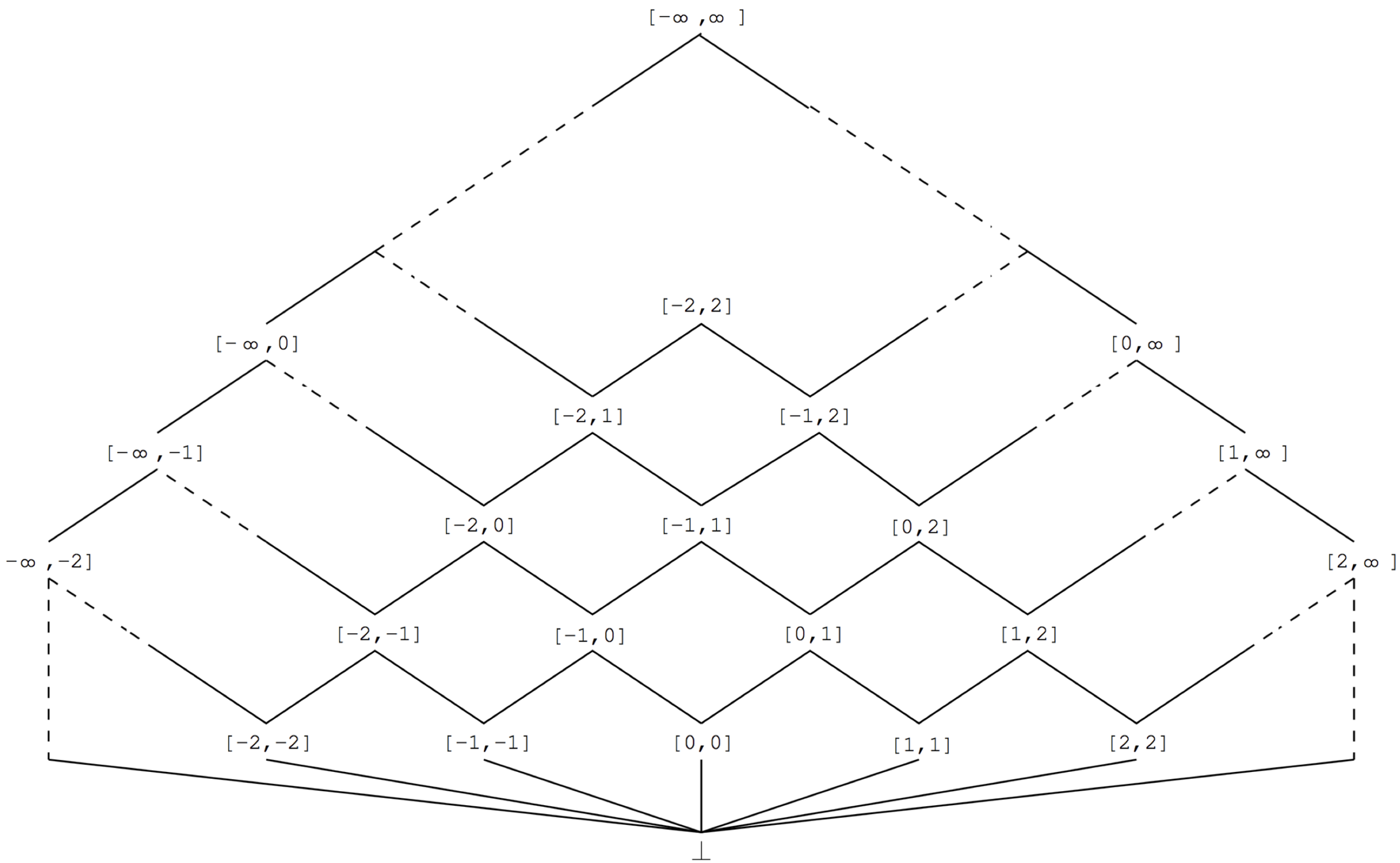
COSE419: Software Verification

Lecture 12 – Interval Analysis

Hakjoo Oh
2024 Spring

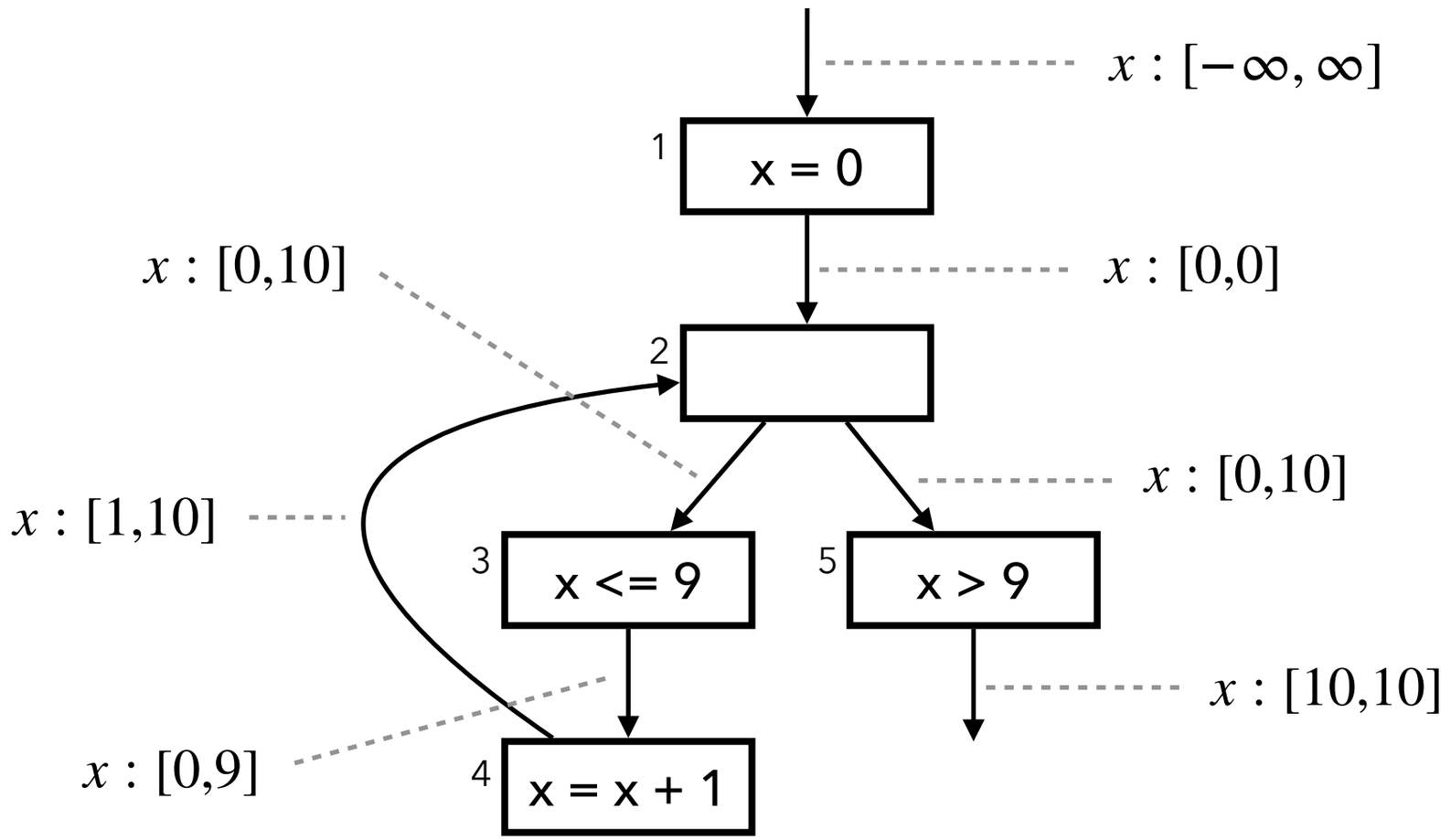
The Interval Domain



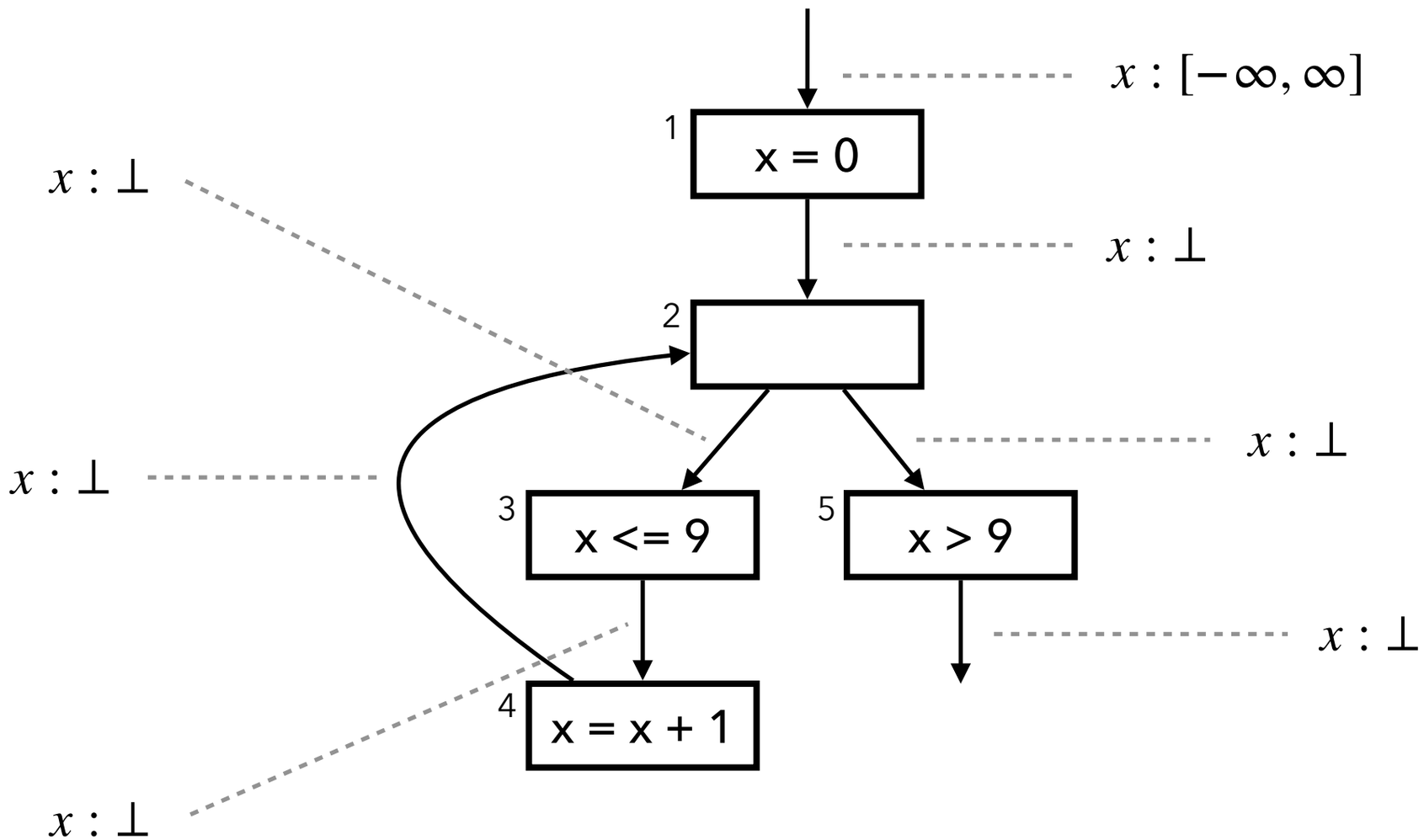
Example Program

```
x = 0;

while (x <= 9)
  x = x + 1;
```

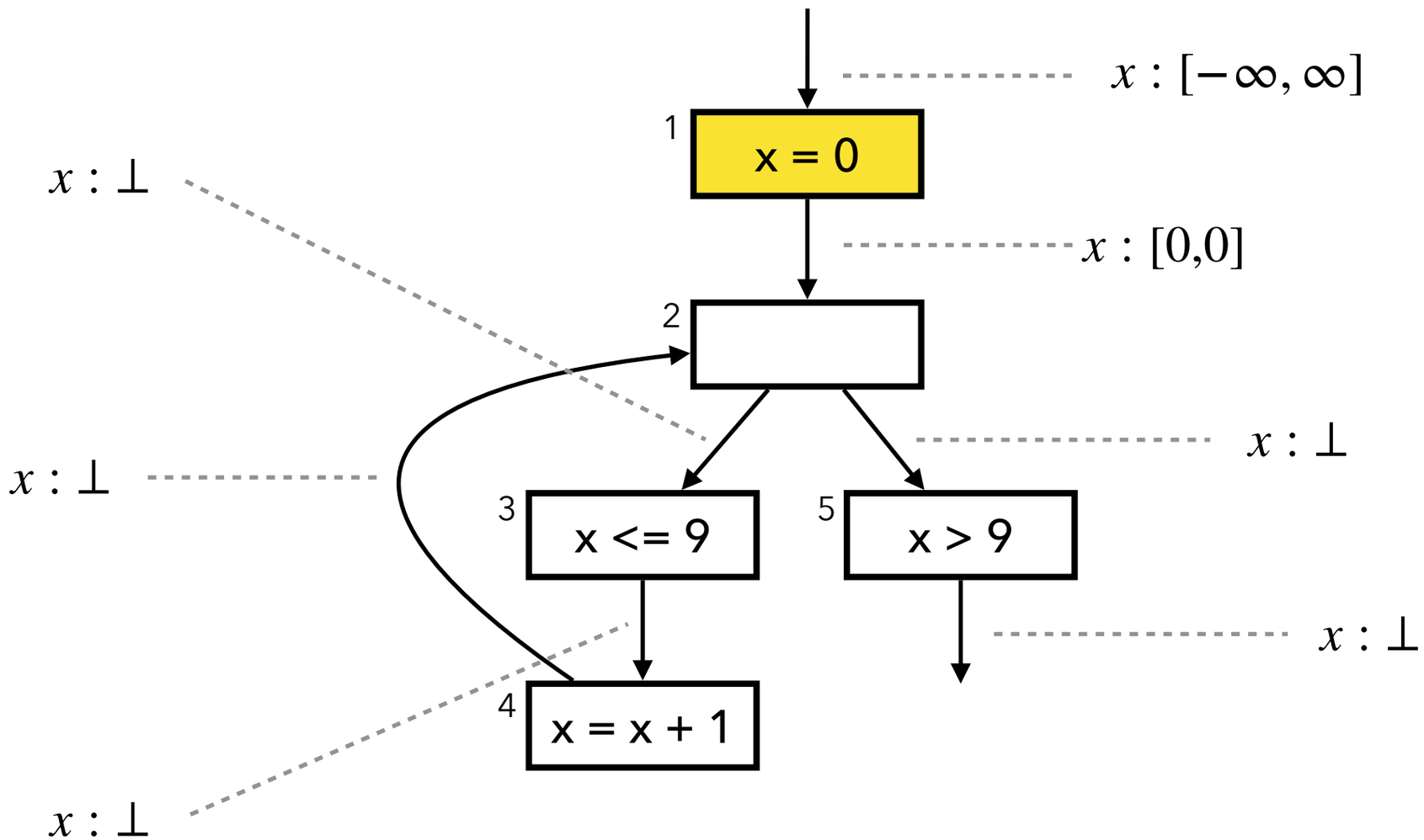


Fixed Point Computation

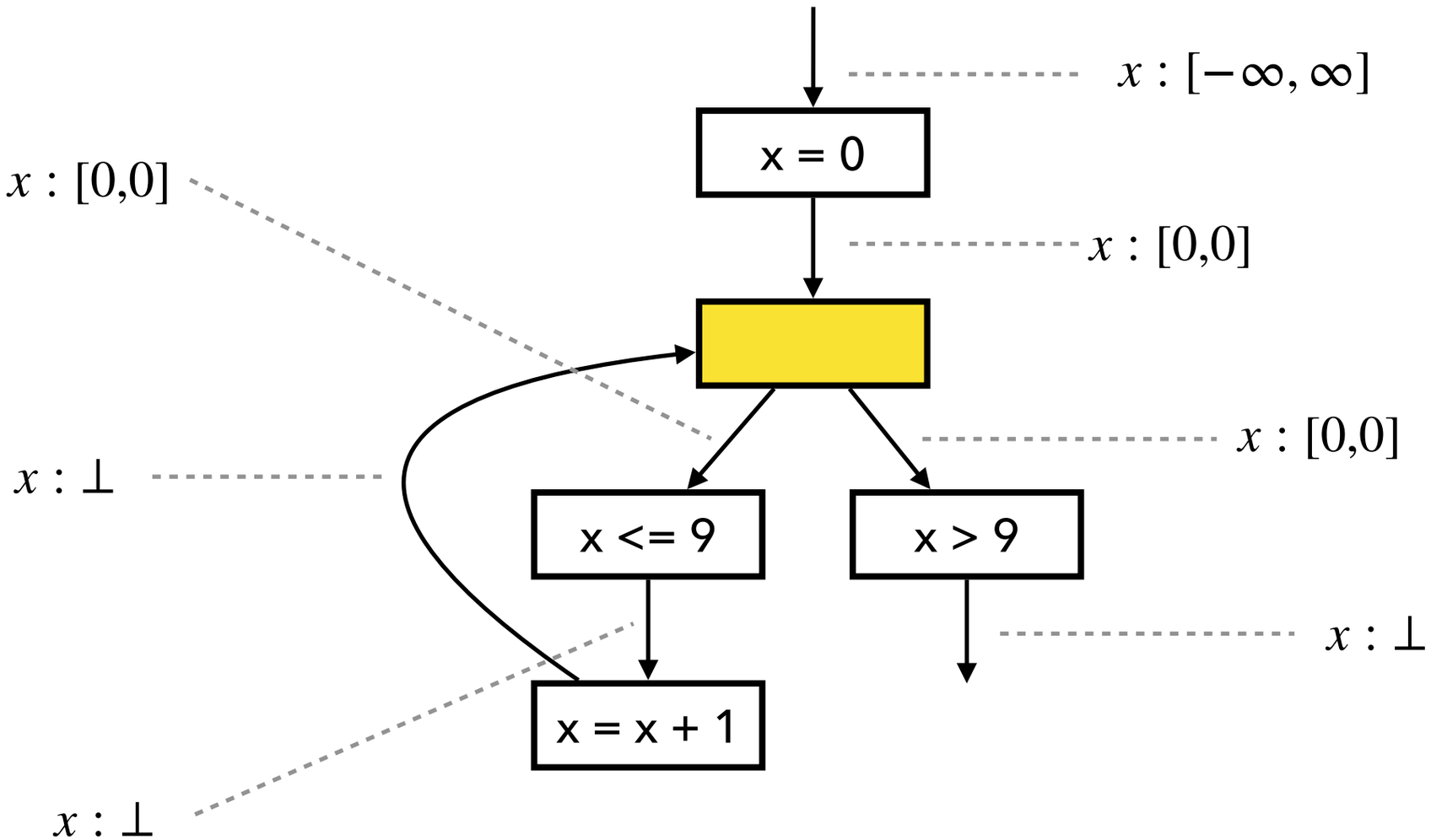


Initial states

Fixed Point Computation

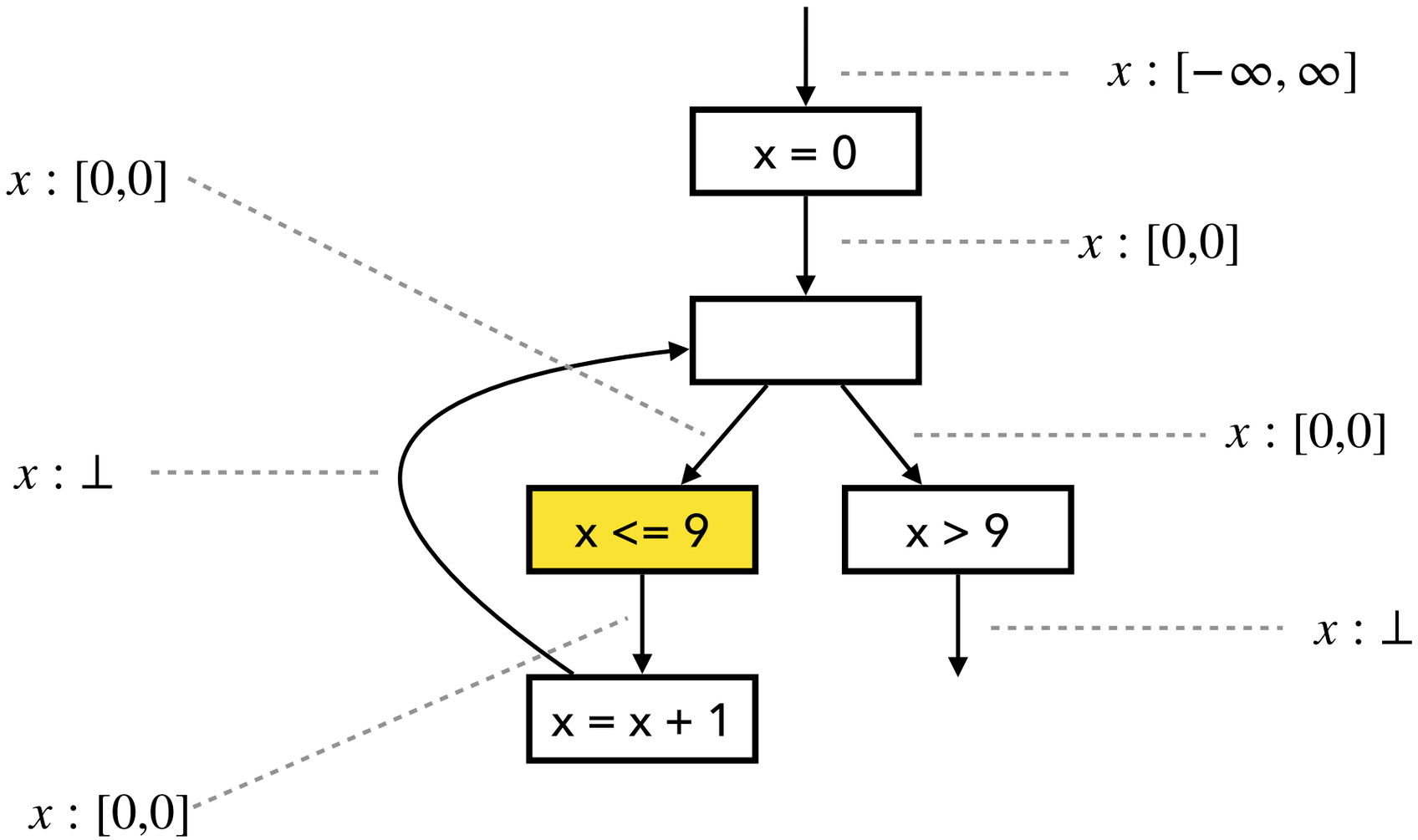


Fixed Point Computation



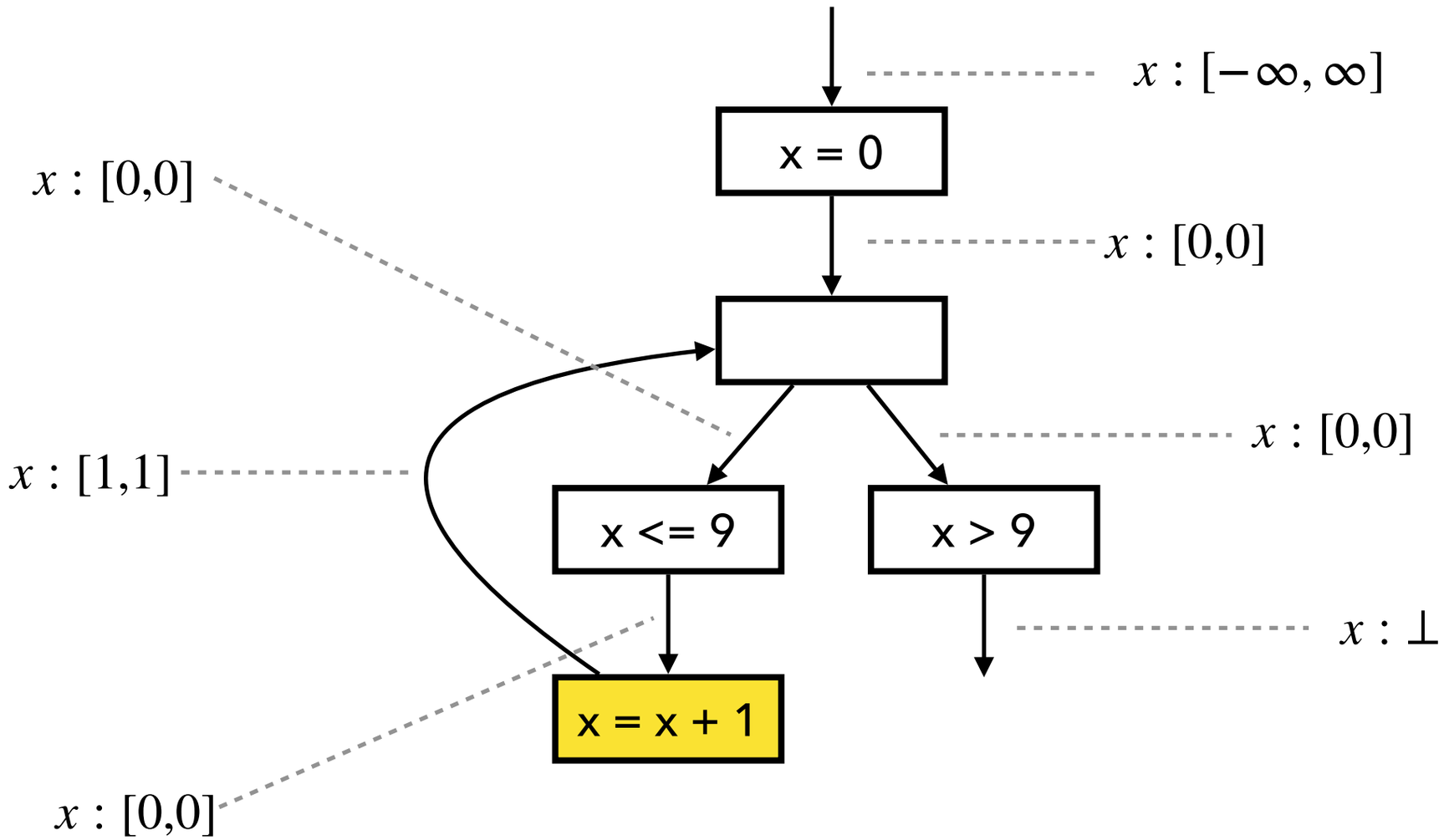
Input state: $[0,0] \sqcup \bot = [0,0]$

Fixed Point Computation

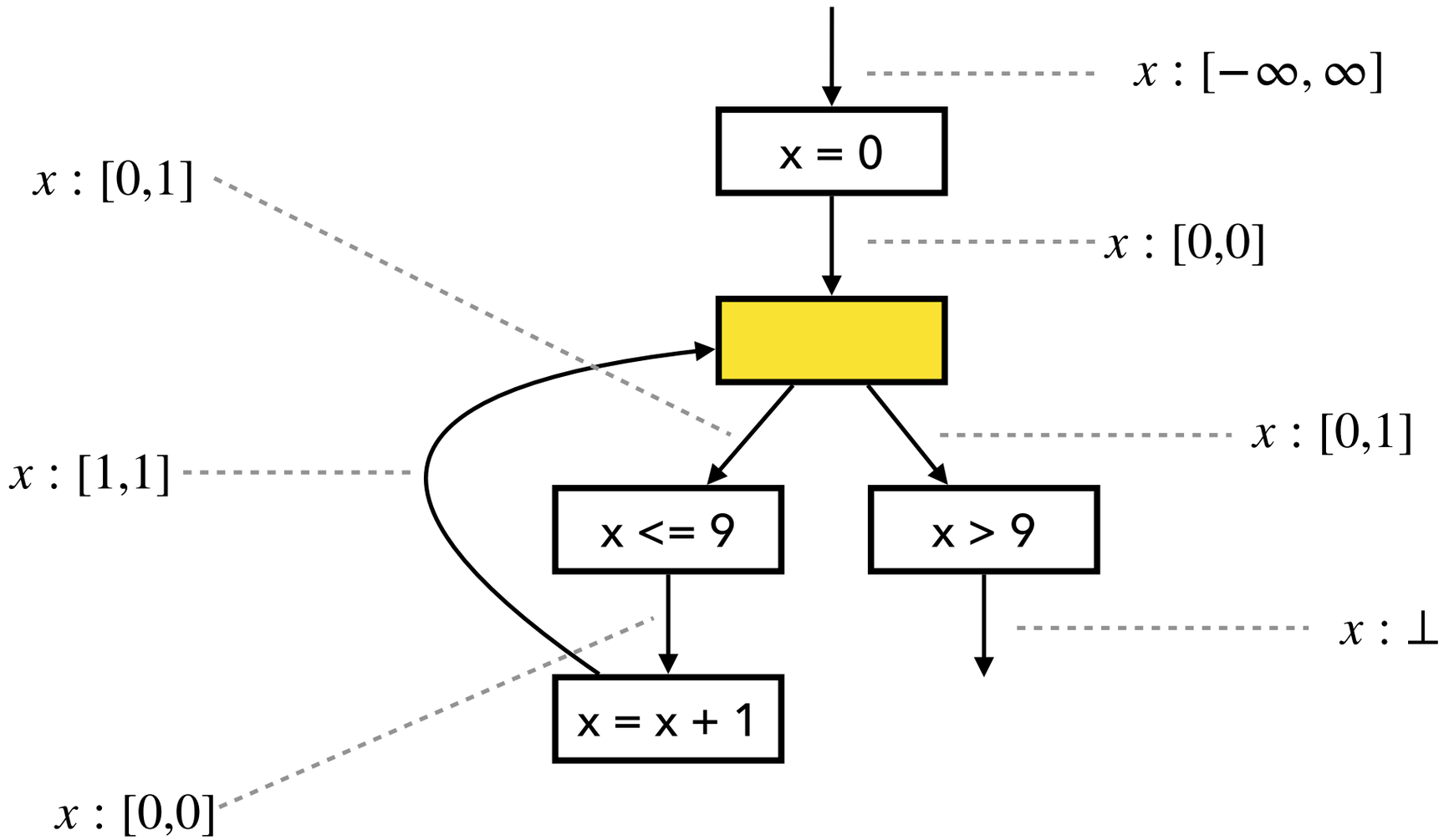


$$[0,0] \sqcap [-\infty,9] = [0,0]$$

Fixed Point Computation



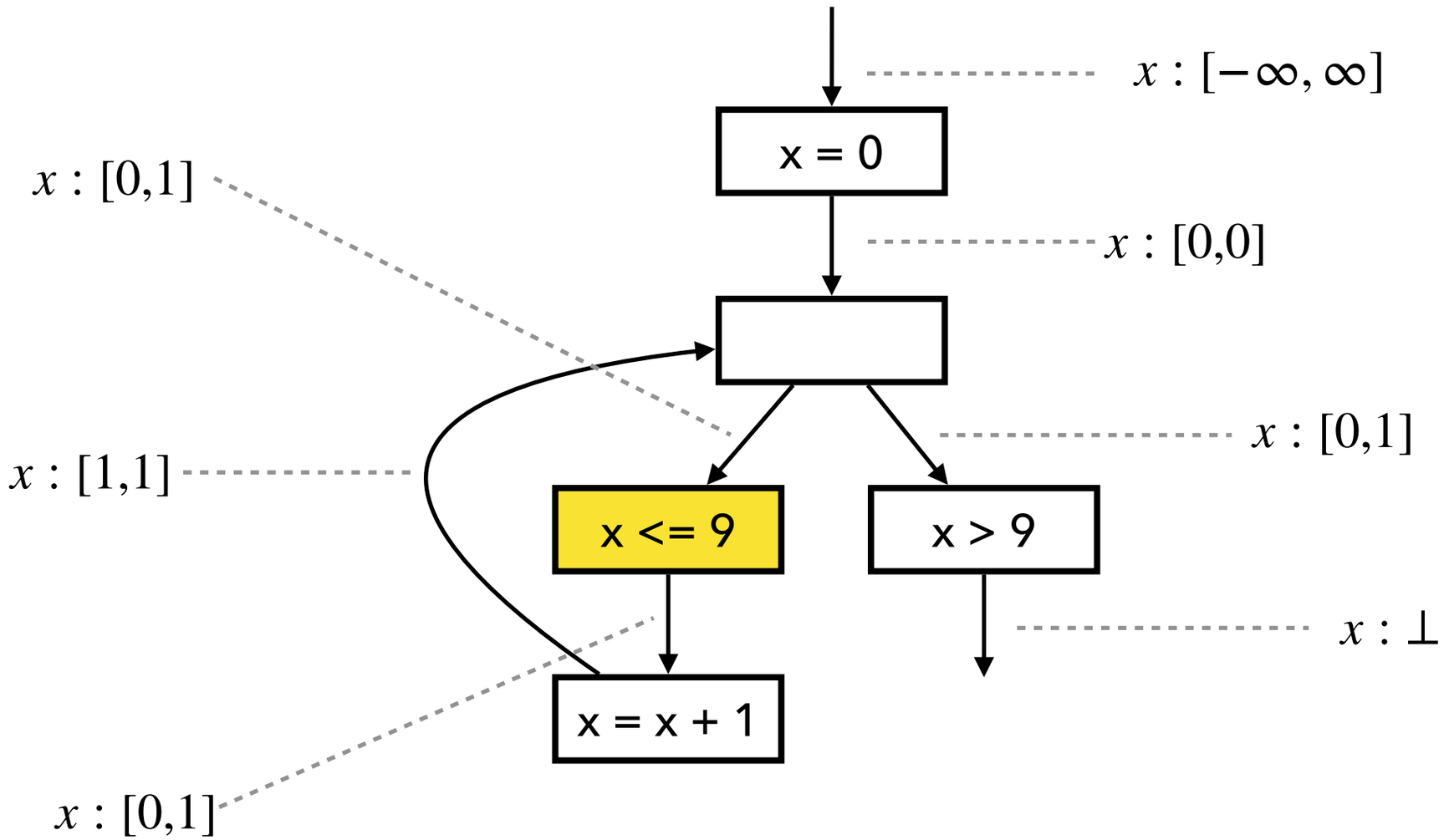
Fixed Point Computation



Input state: $[0,0] \sqcup [1,1] = [0,1]$

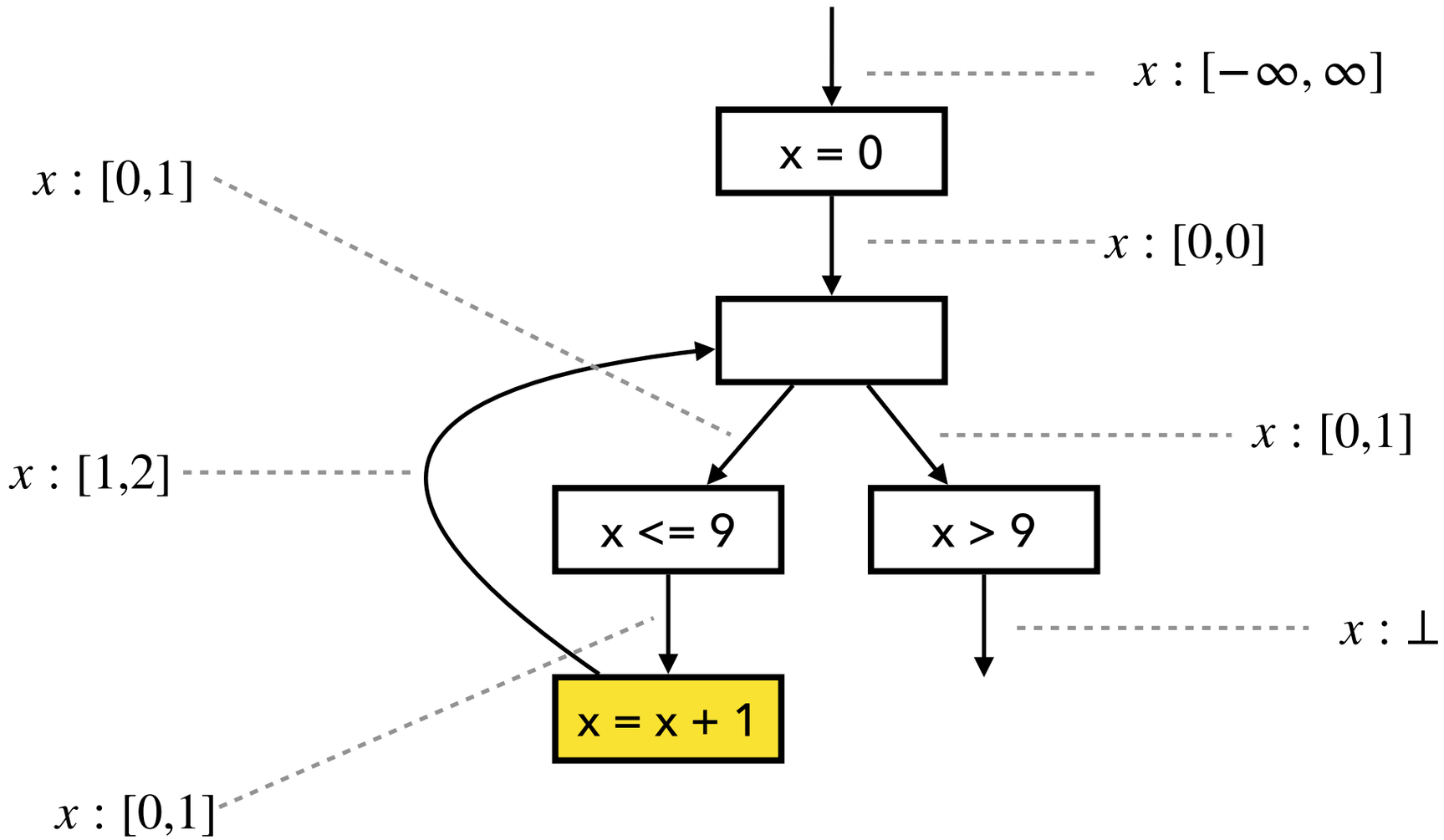
(1st iteration of loop)

Fixed Point Computation

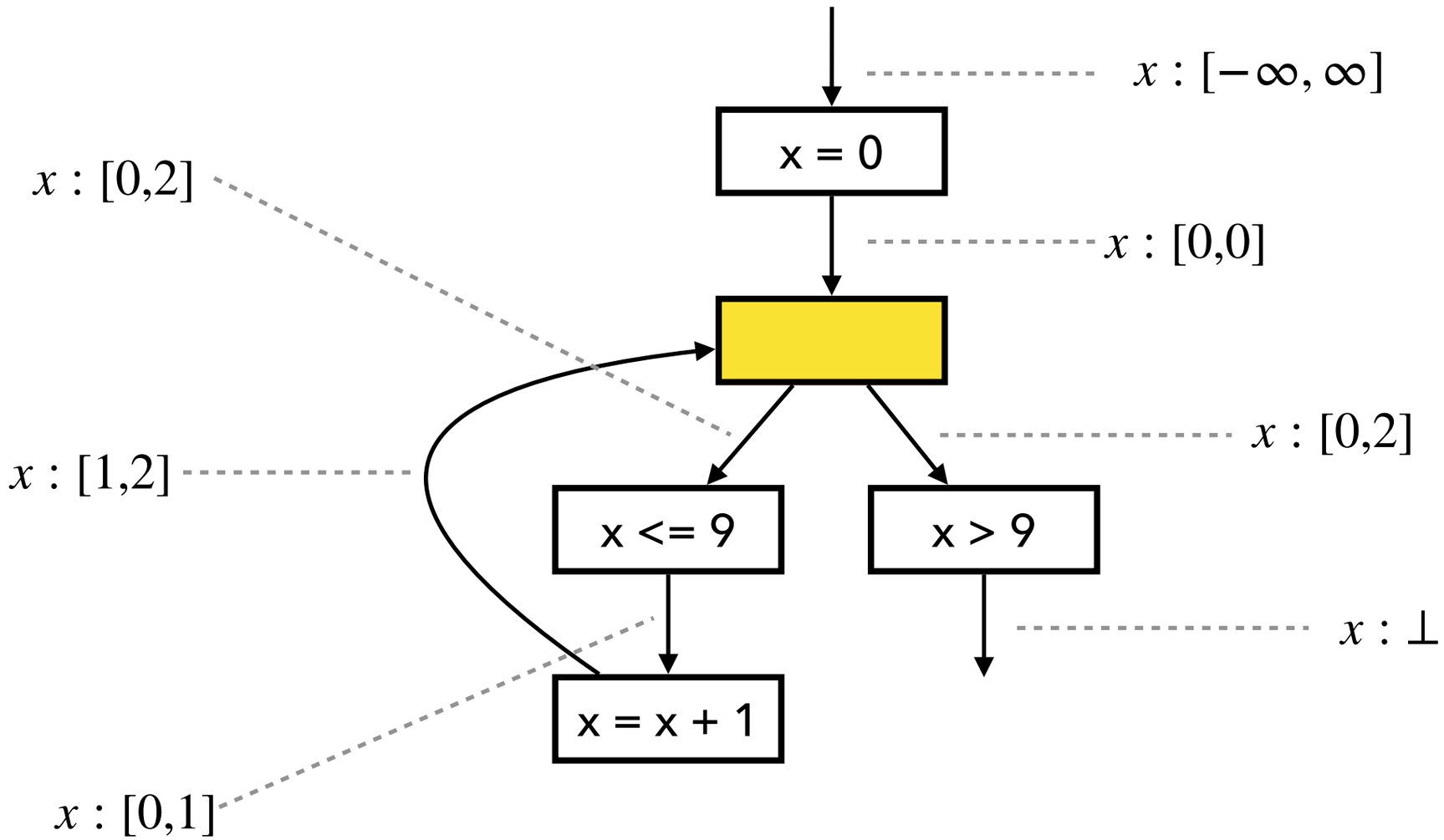


$$[0,1] \sqcap [-\infty,9] = [0,1]$$

Fixed Point Computation



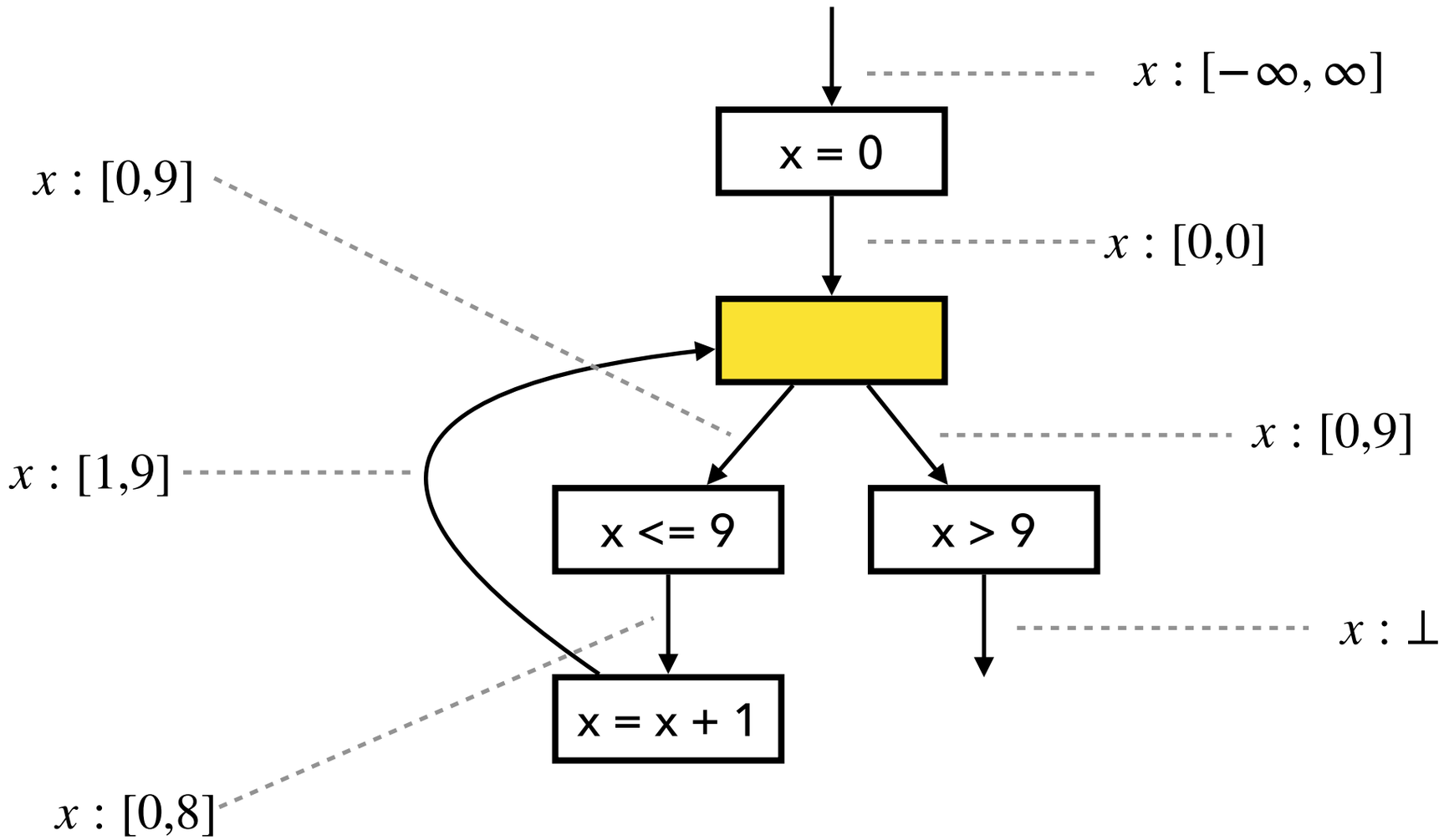
Fixed Point Computation



Input state: $[0,0] \sqcup [1,2] = [0,2]$

(2nd iteration of loop)

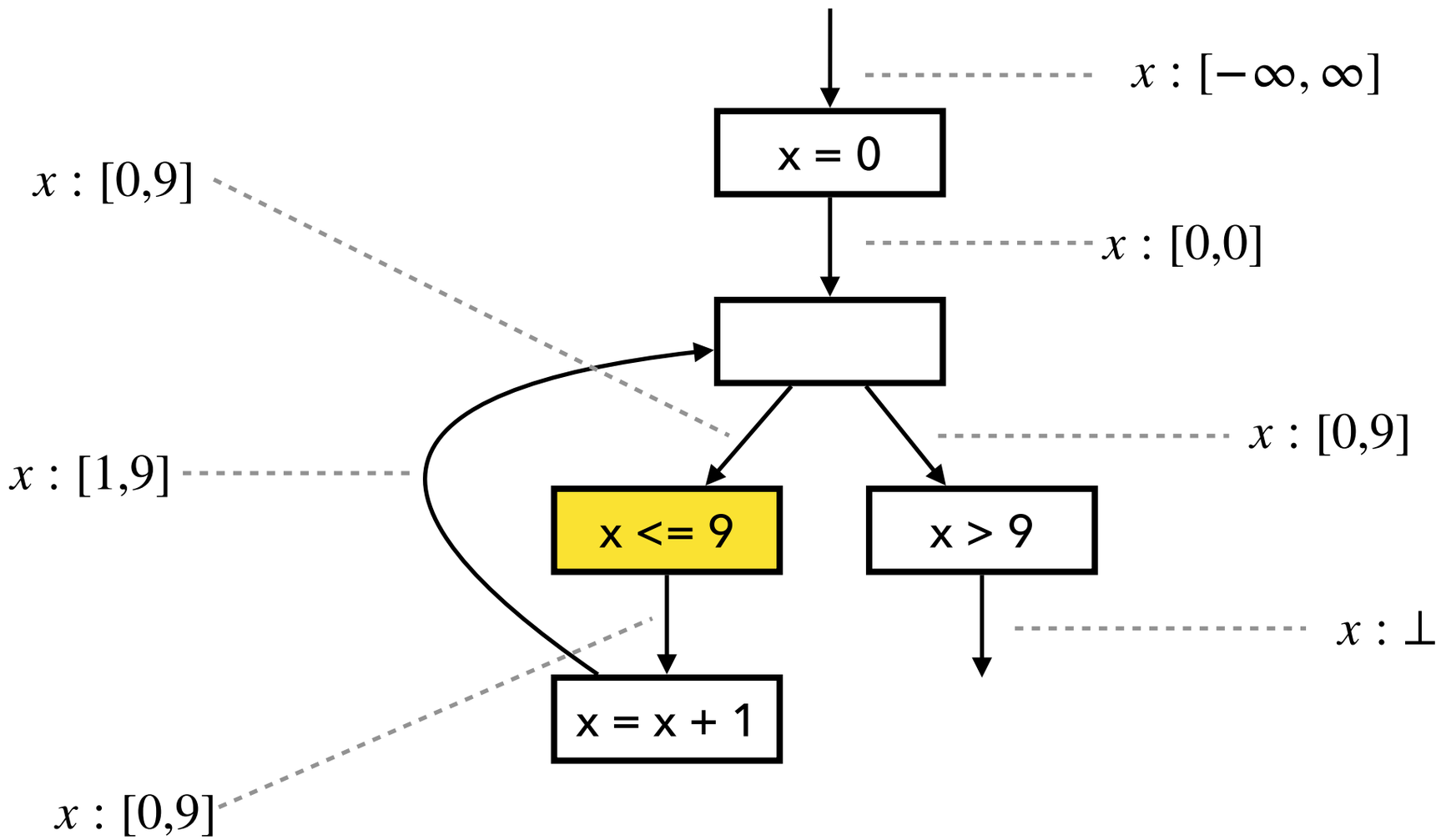
Fixed Point Computation



Input state: $[0,0] \sqcup [1,9] = [0,9]$

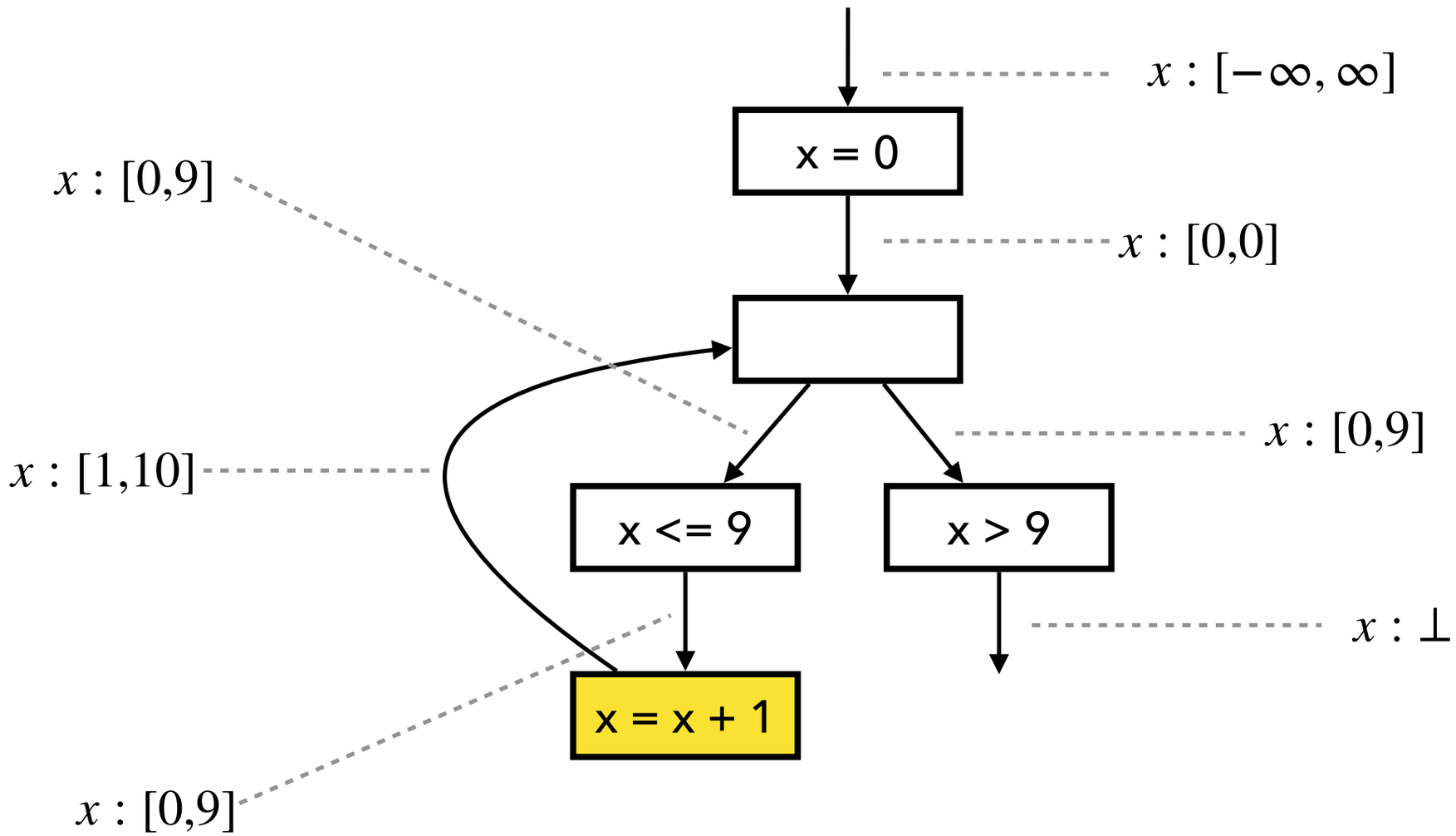
(9th iteration of loop)

Fixed Point Computation

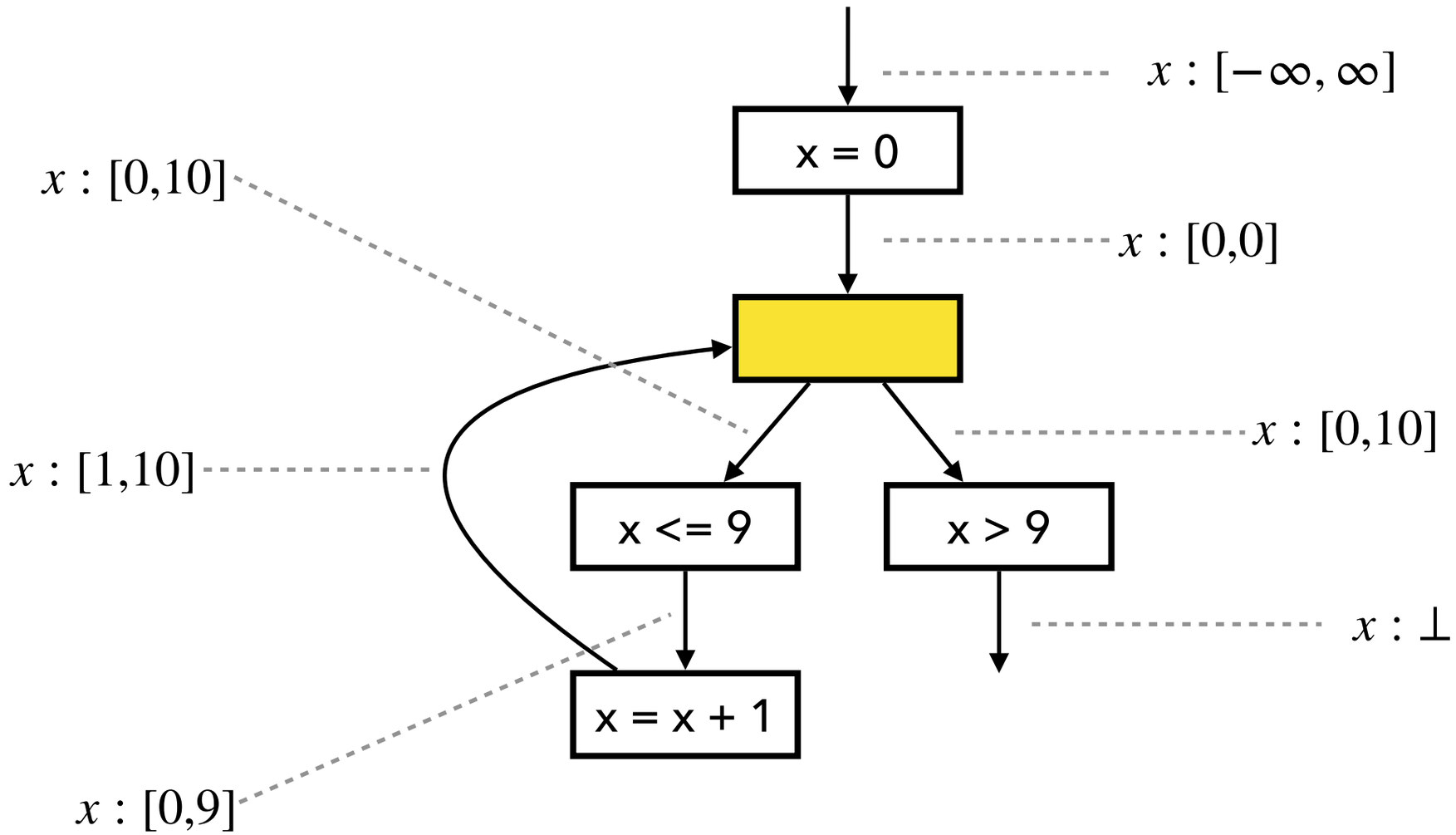


$$[0,9] \sqcap [-\infty,9] = [0,9]$$

Fixed Point Computation



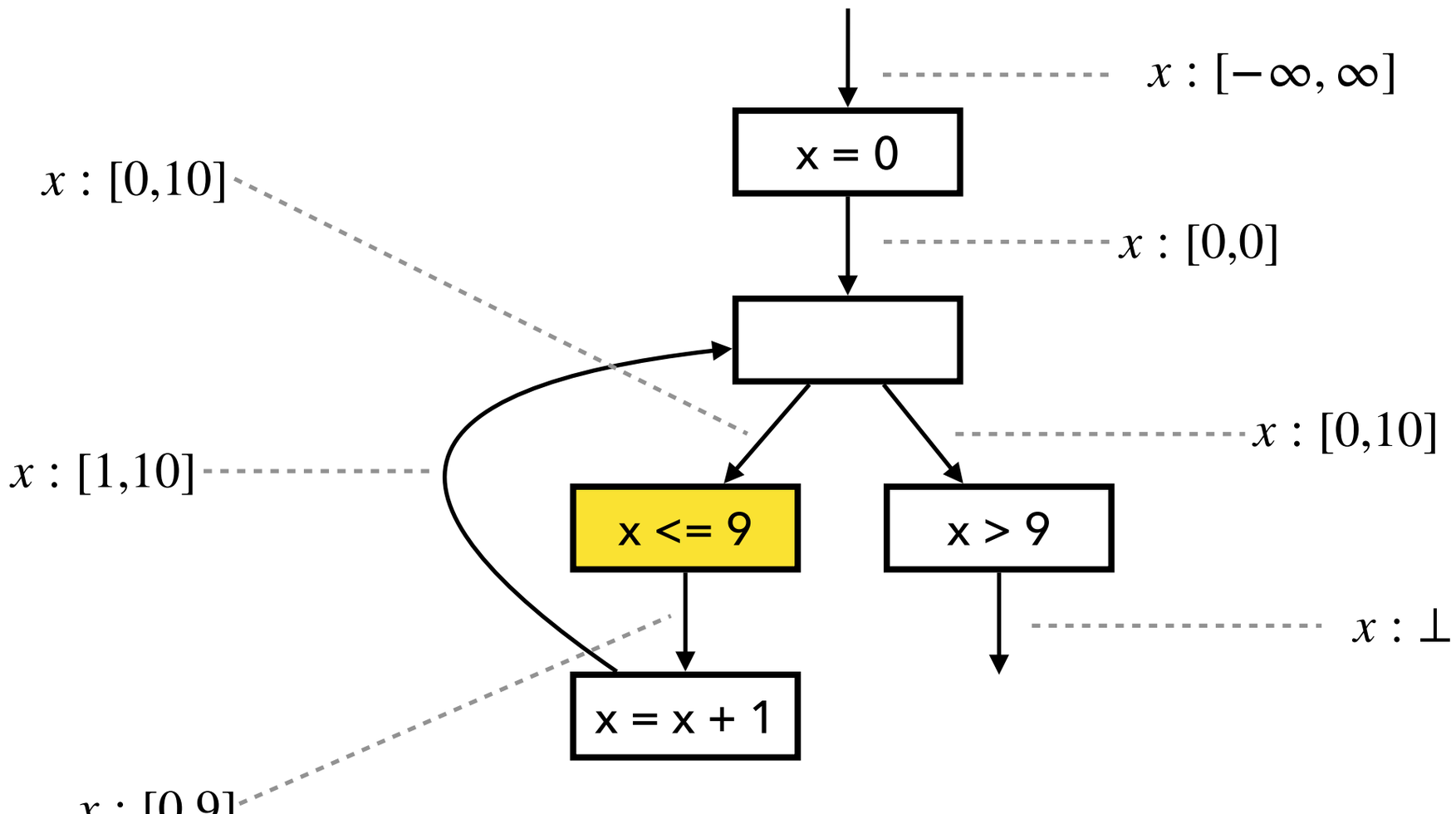
Fixed Point Computation



Input state: $[0,0] \sqcup [1,10] = [0,10]$

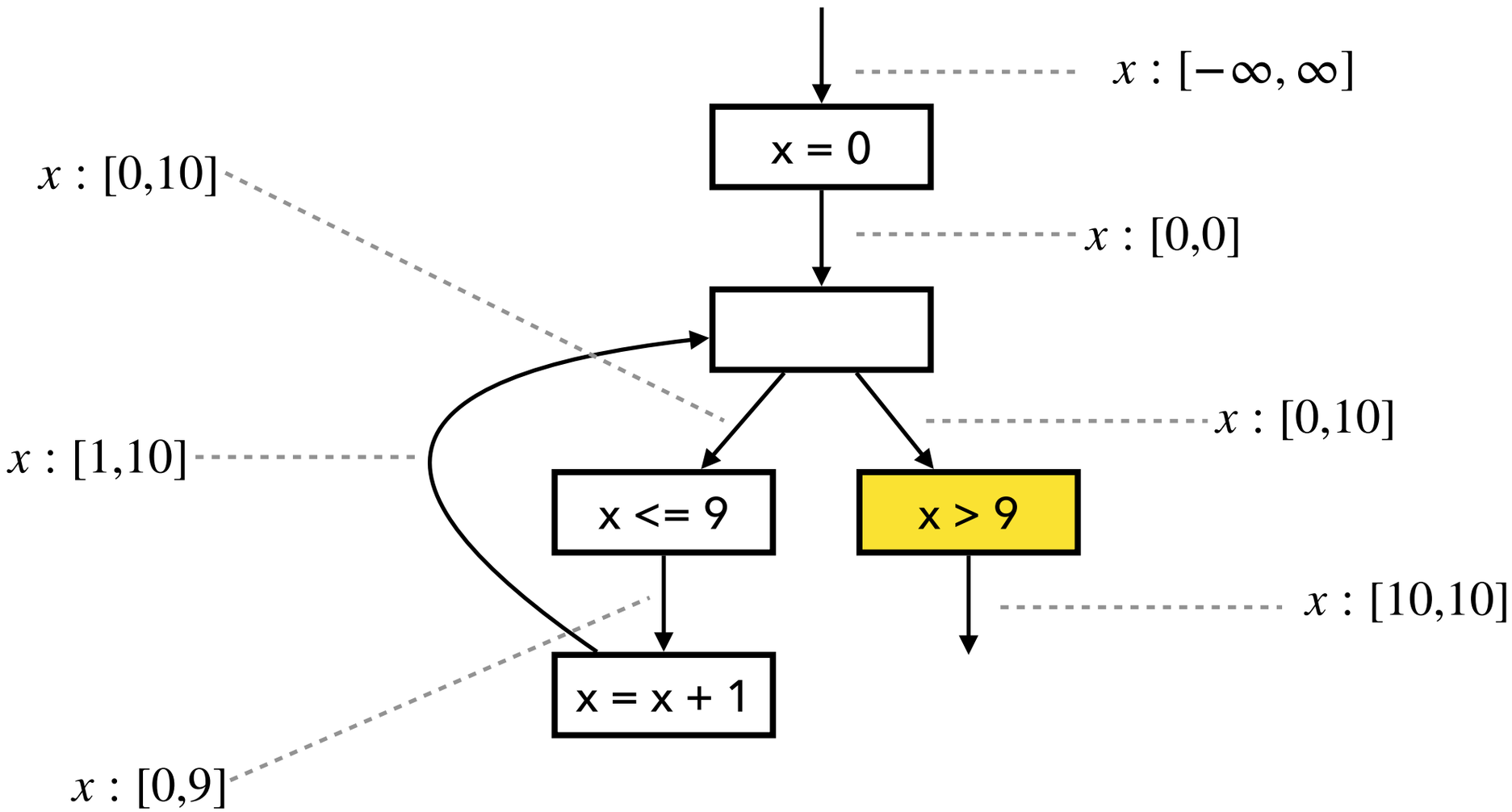
(10th iteration of loop)

Fixed Point Computation



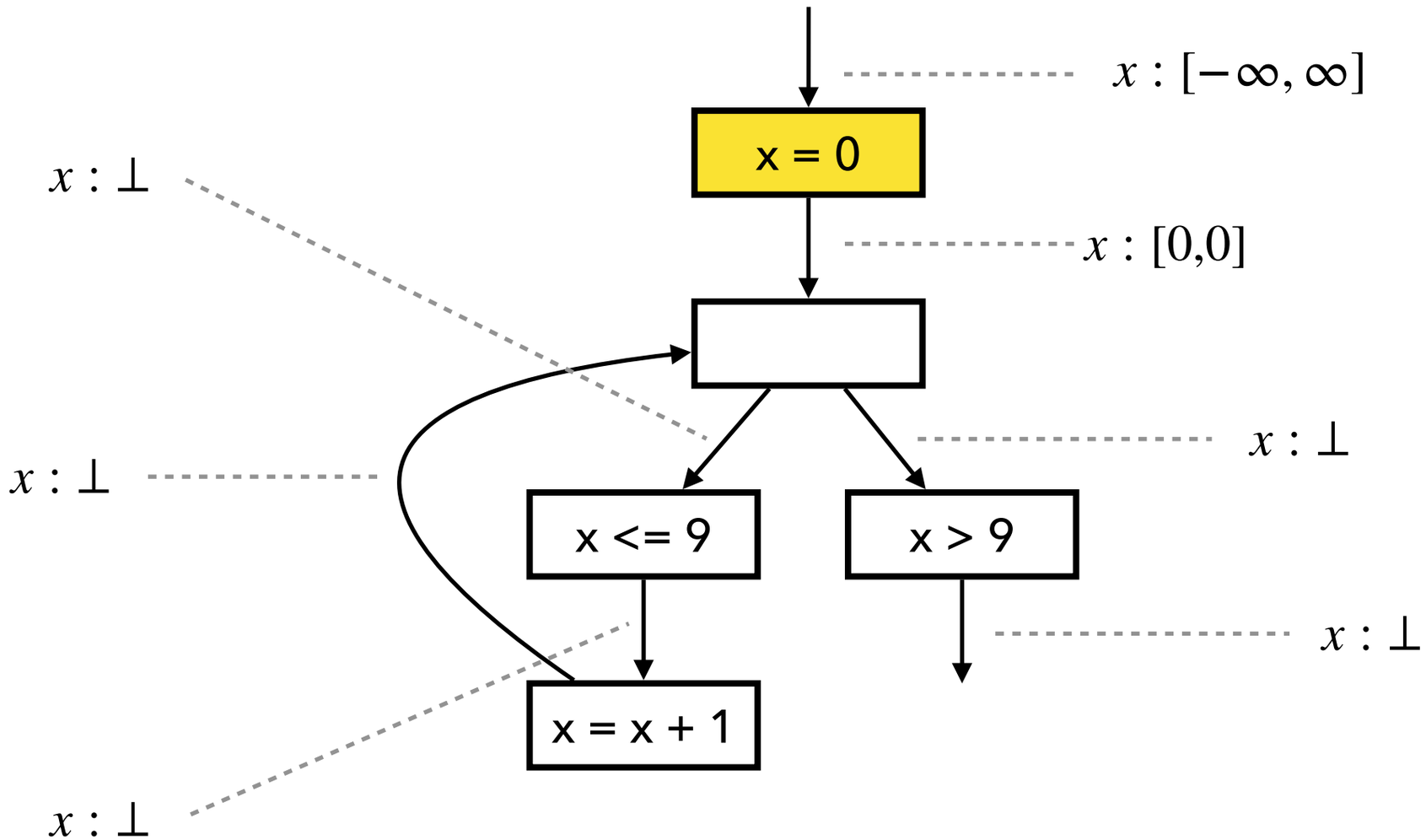
$$[0,10] \sqcap [-\infty,9] = [0,9]$$

Fixed Point Computation

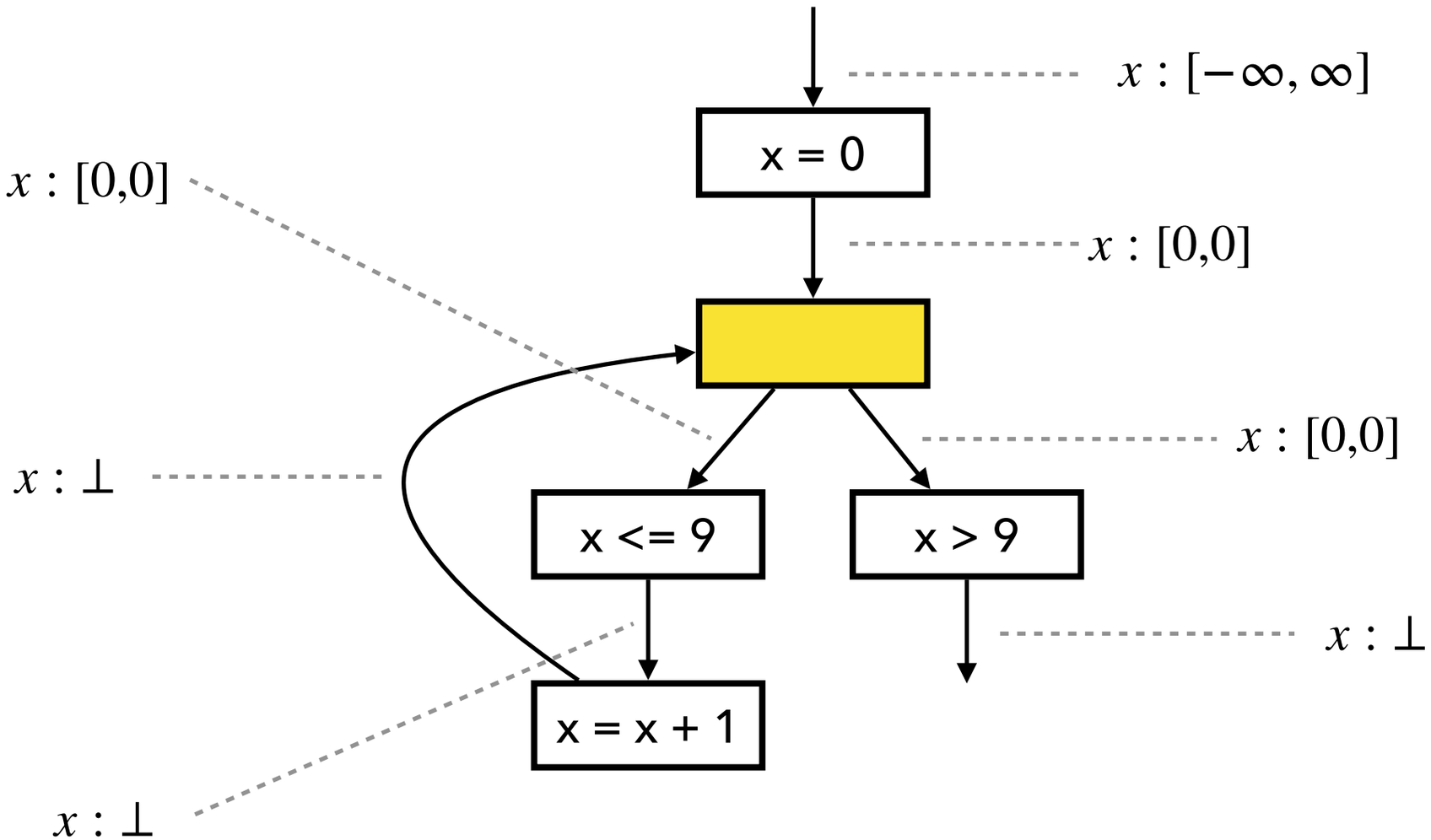


$$[0,10] \sqcap [10,\infty] = [10,10]$$

Fixed Point Comp. with Widening

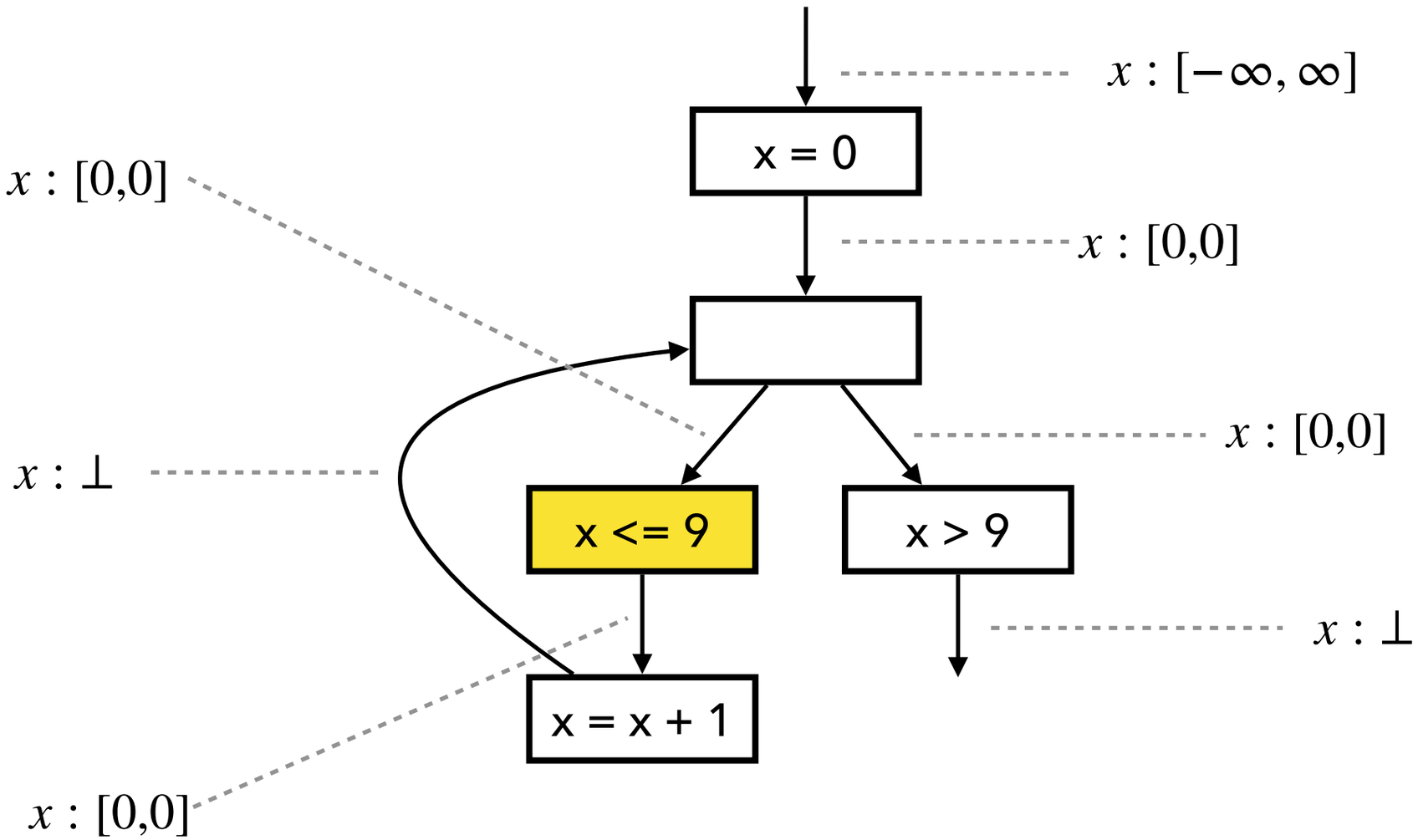


Fixed Point Comp. with Widening



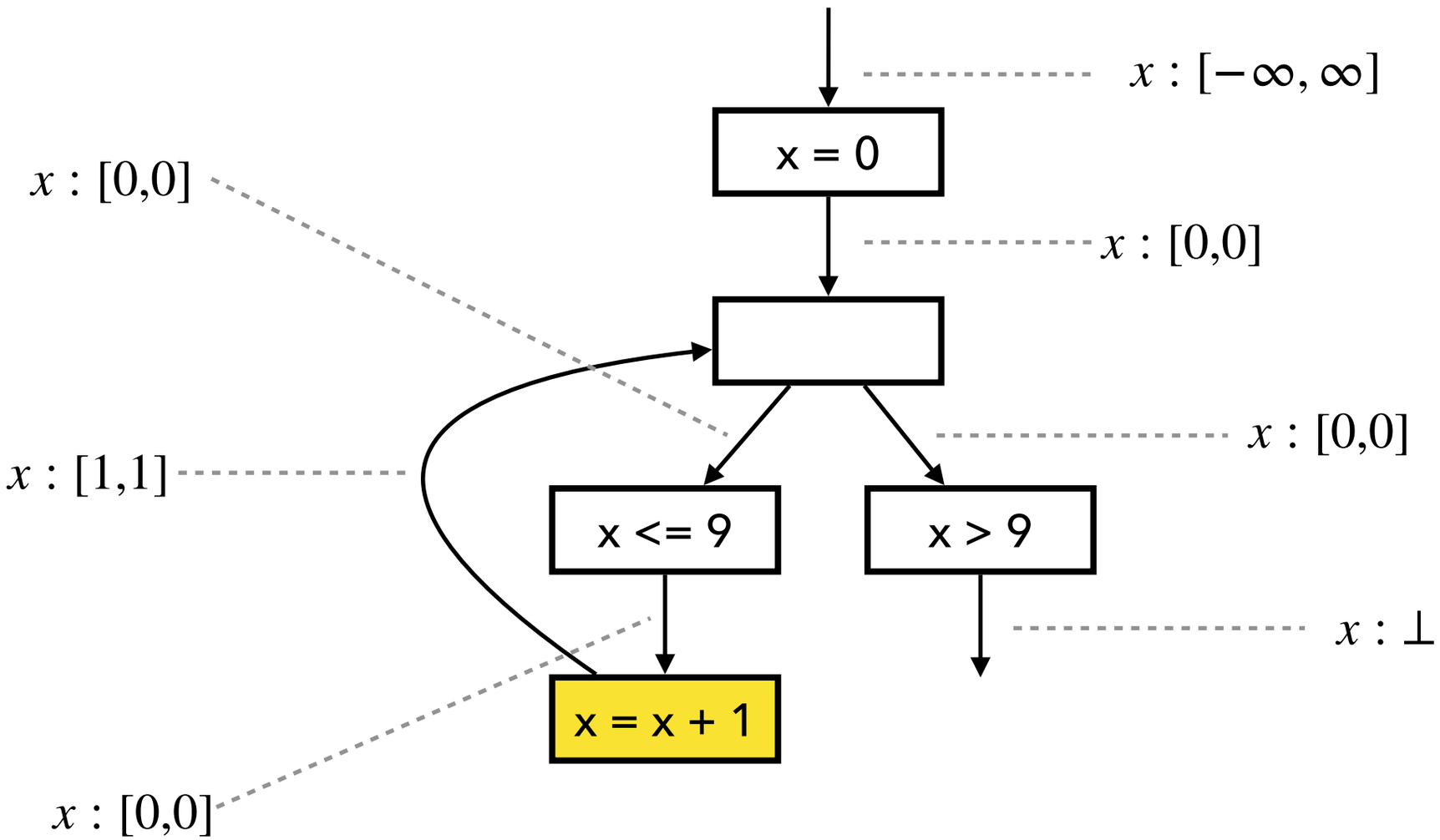
Input state: $[0,0] \sqcup \bot = [0,0]$

Fixed Point Comp. with Widening



$$[0,0] \sqcap [-\infty,9] = [0,0]$$

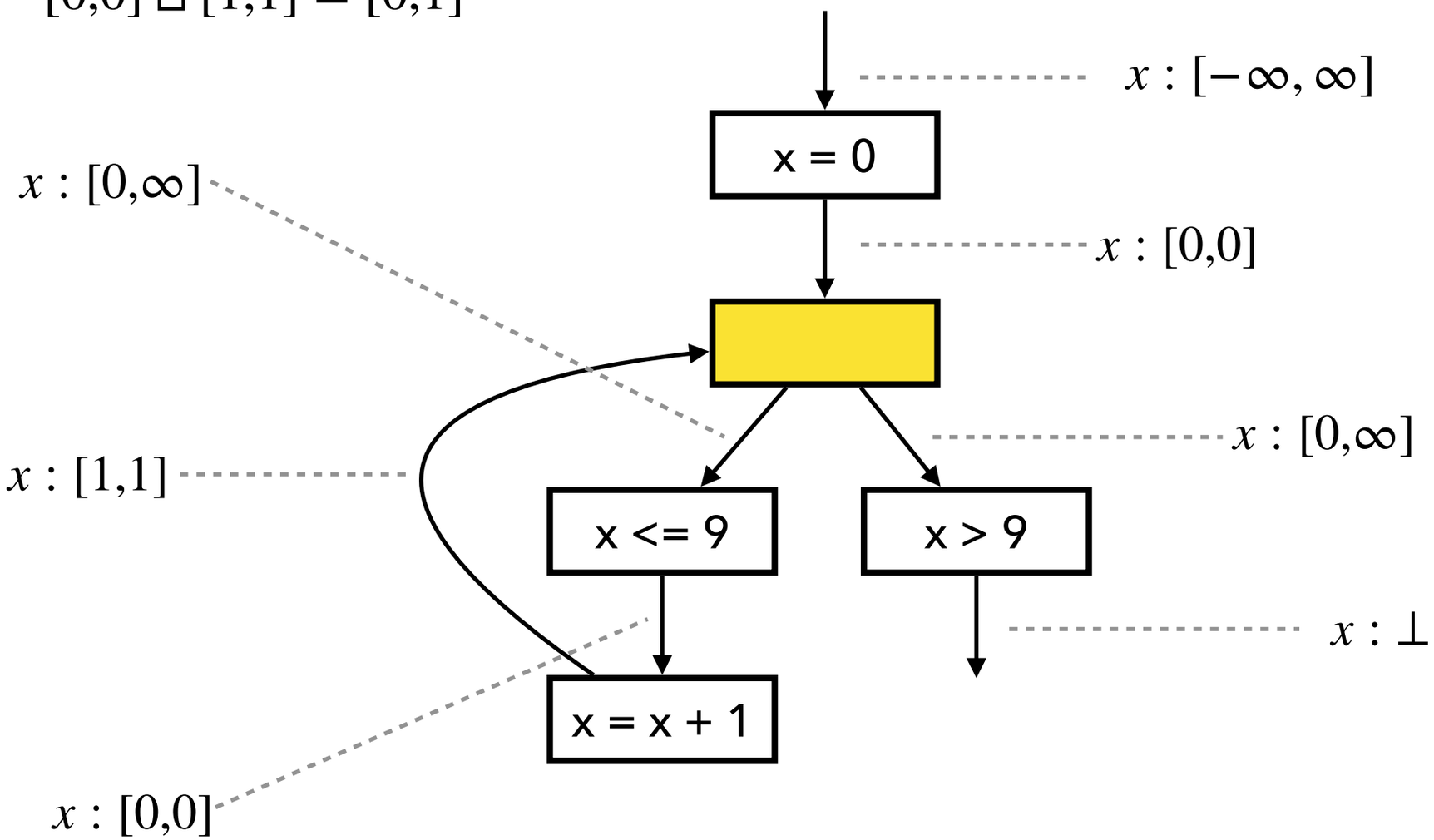
Fixed Point Comp. with Widening



Fixed Point Comp. with Widening

1. Compute output by joining inputs:

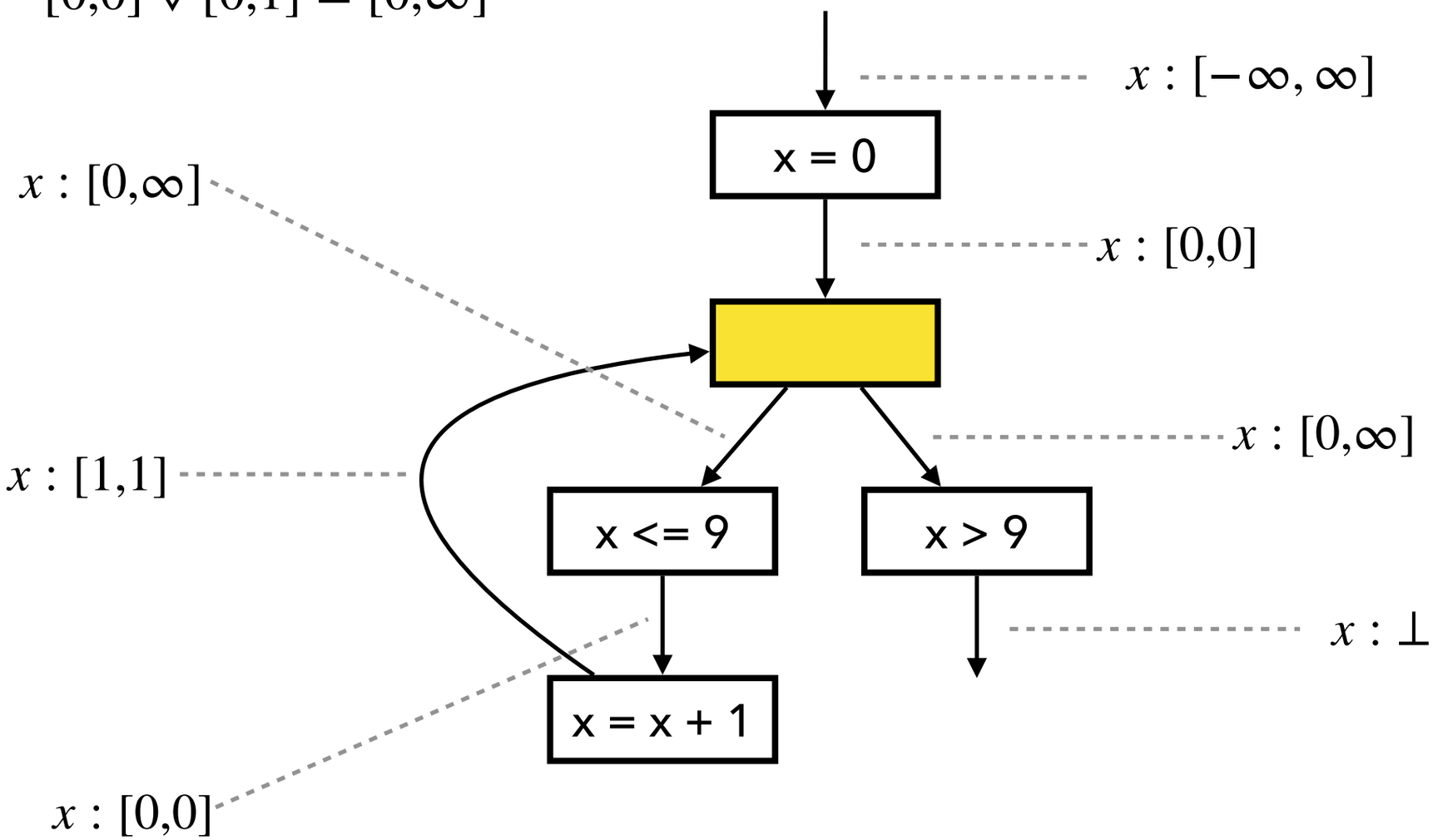
$$[0,0] \sqcup [1,1] = [0,1]$$



Fixed Point Comp. with Widening

2. Apply widening with old output:

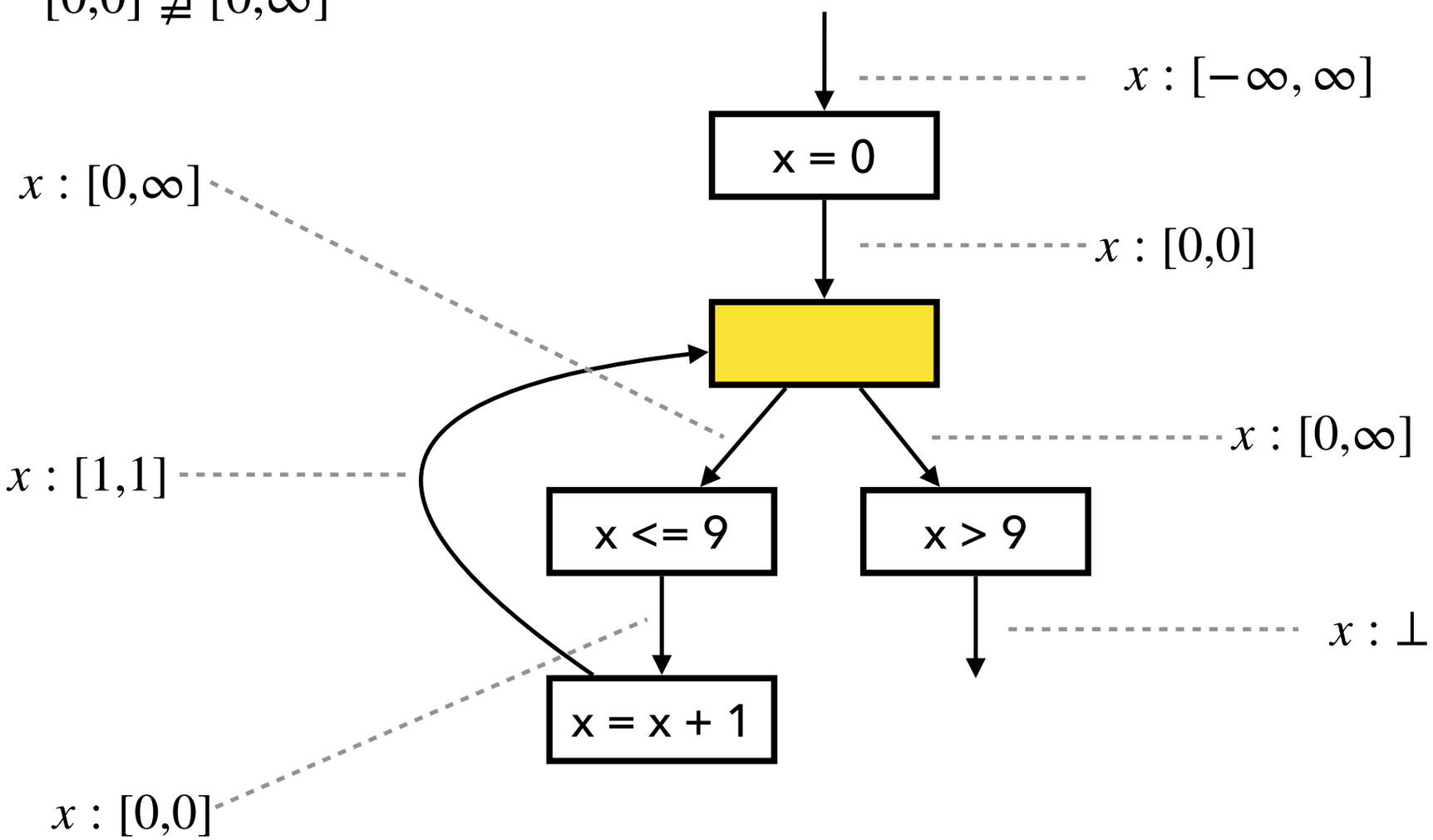
$$[0,0] \triangledown [0,1] = [0,\infty]$$



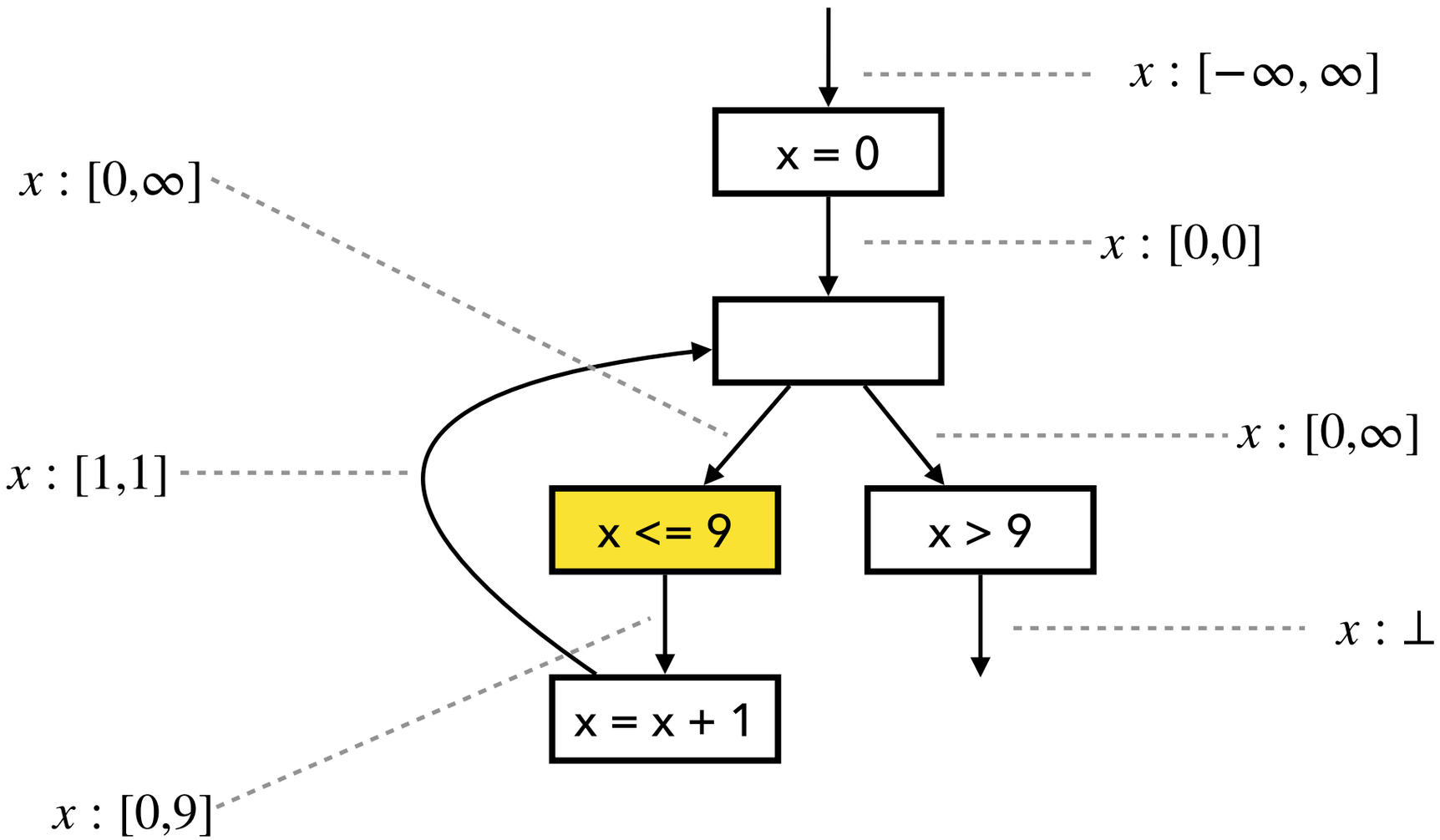
Fixed Point Comp. with Widening

3. Check if fixed point is reached

$[0,0] \not\supseteq [0,\infty]$

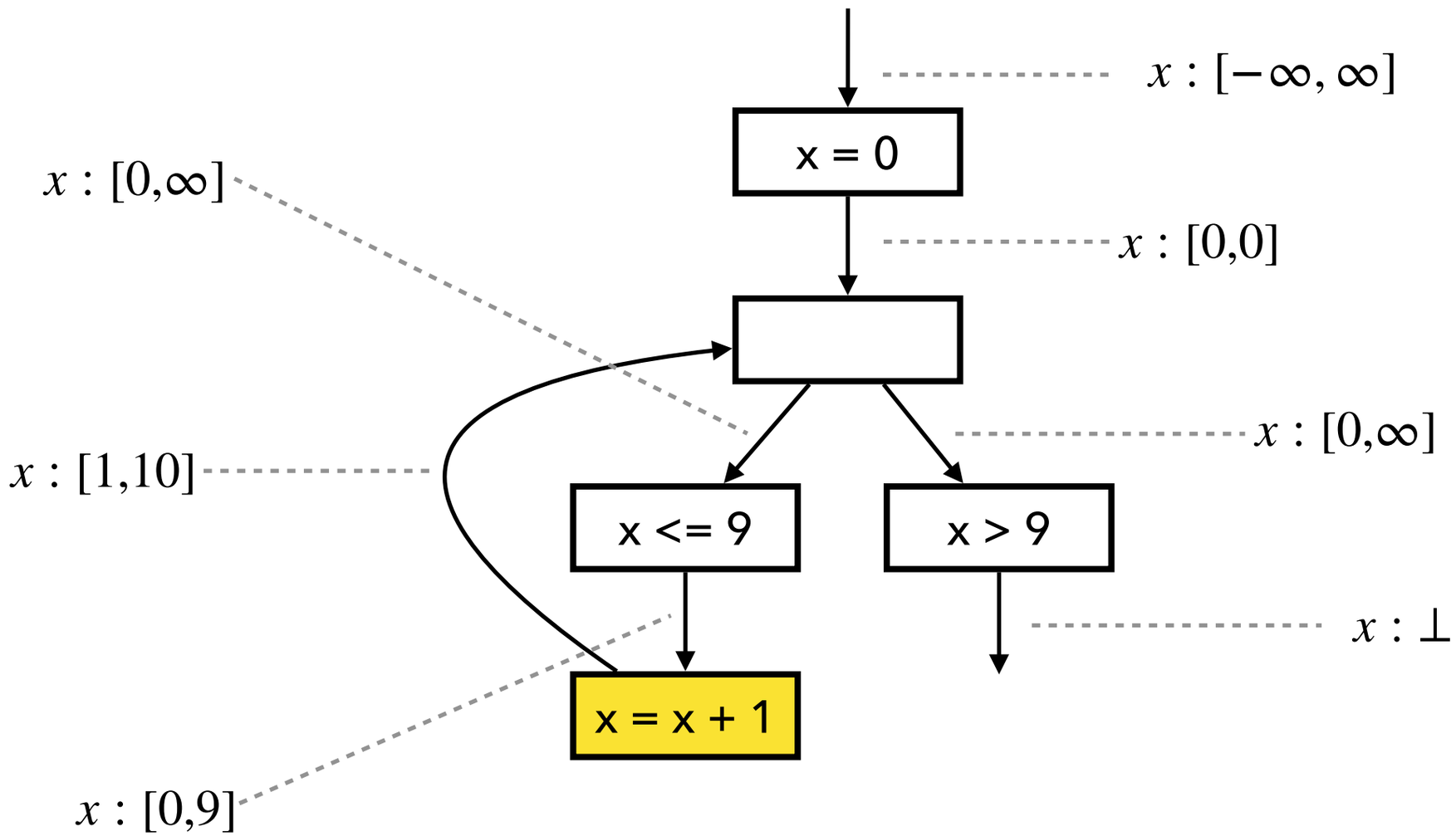


Fixed Point Comp. with Widening



$$[0,\infty] \sqcap [-\infty,9] = [0,9]$$

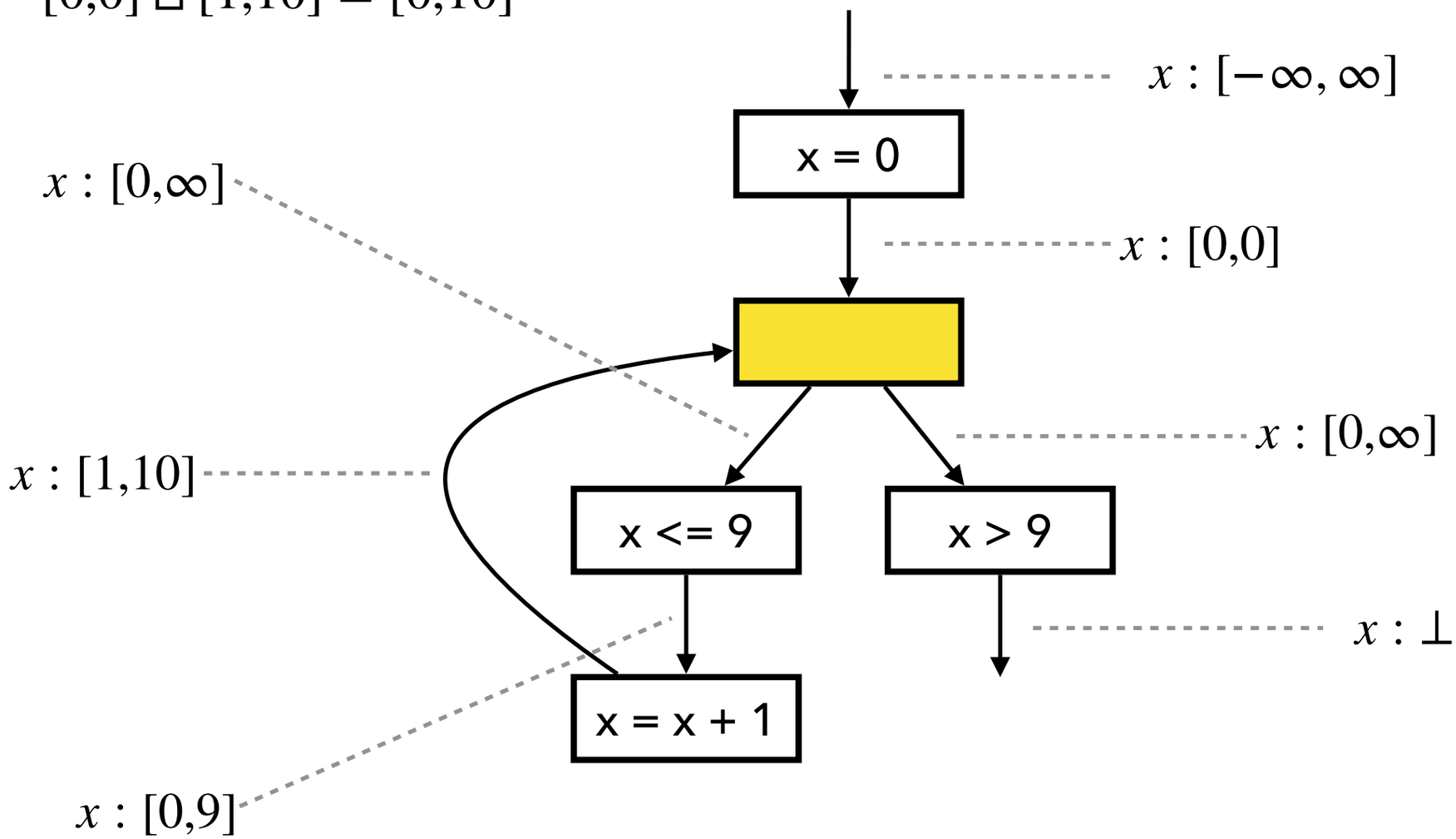
Fixed Point Comp. with Widening



Fixed Point Comp. with Widening

1. Compute output by joining inputs:

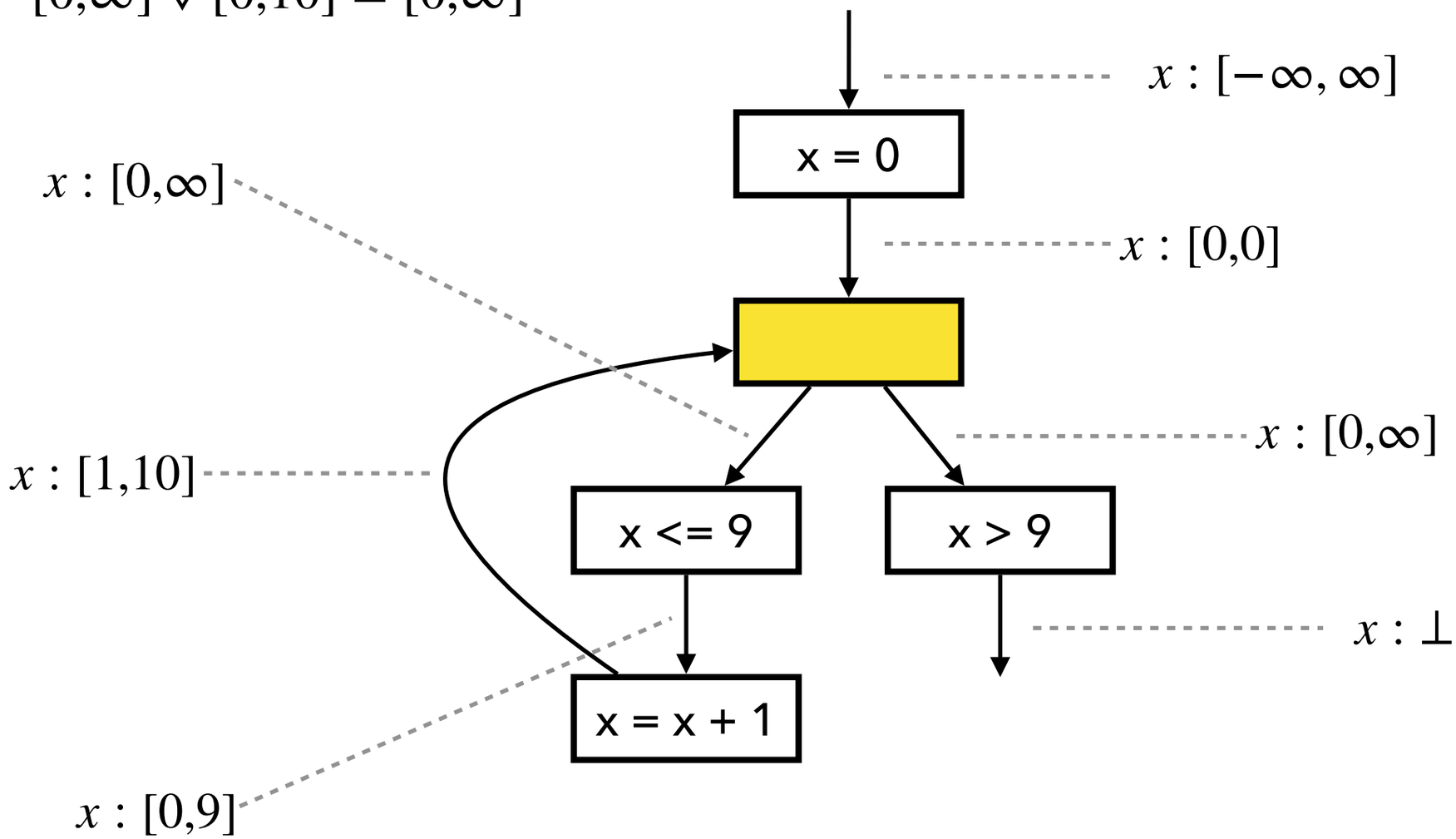
$$[0,0] \sqcup [1,10] = [0,10]$$



Fixed Point Comp. with Widening

2. Apply widening with old output:

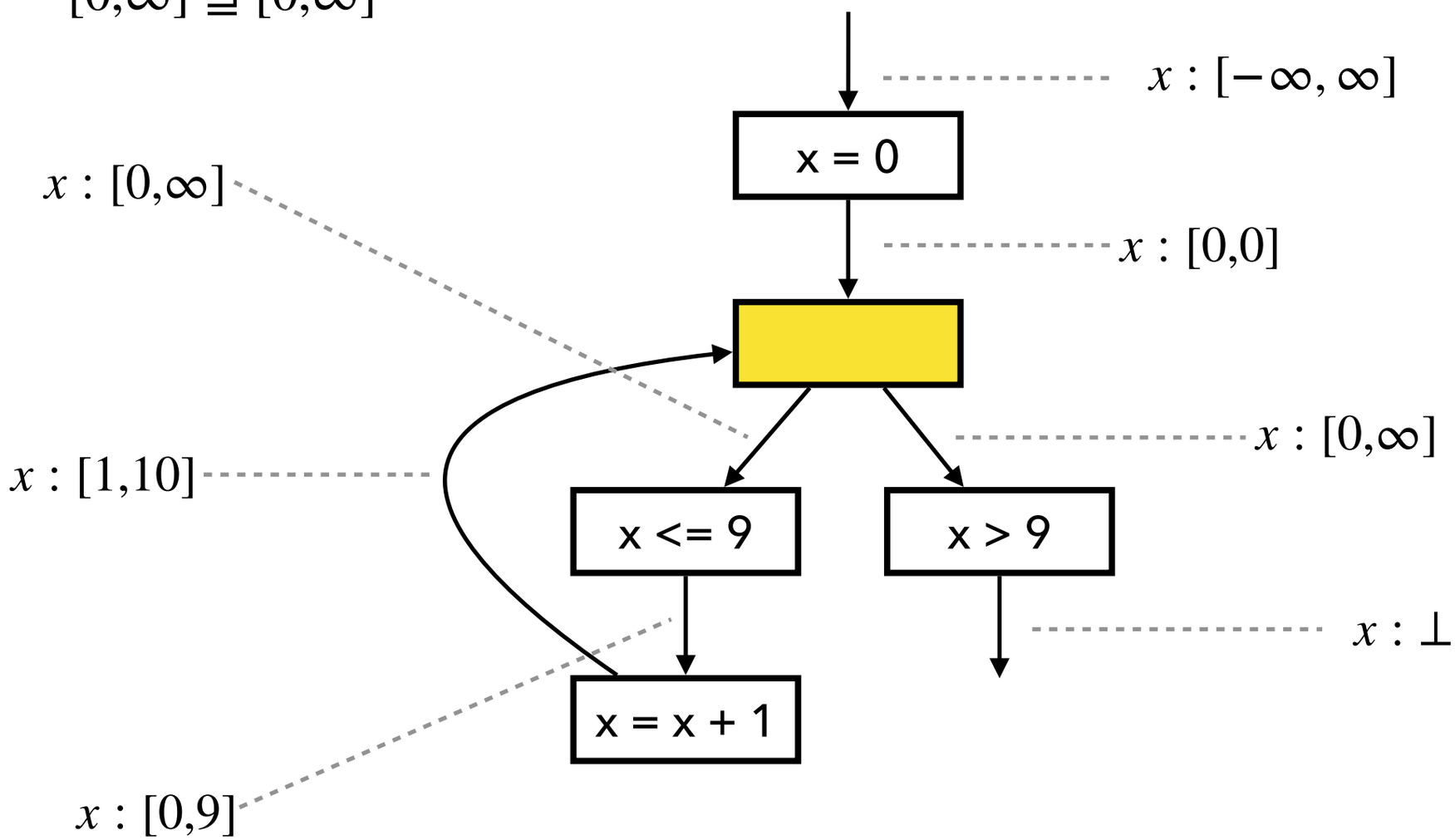
$$[0,\infty] \triangledown [0,10] = [0,\infty]$$



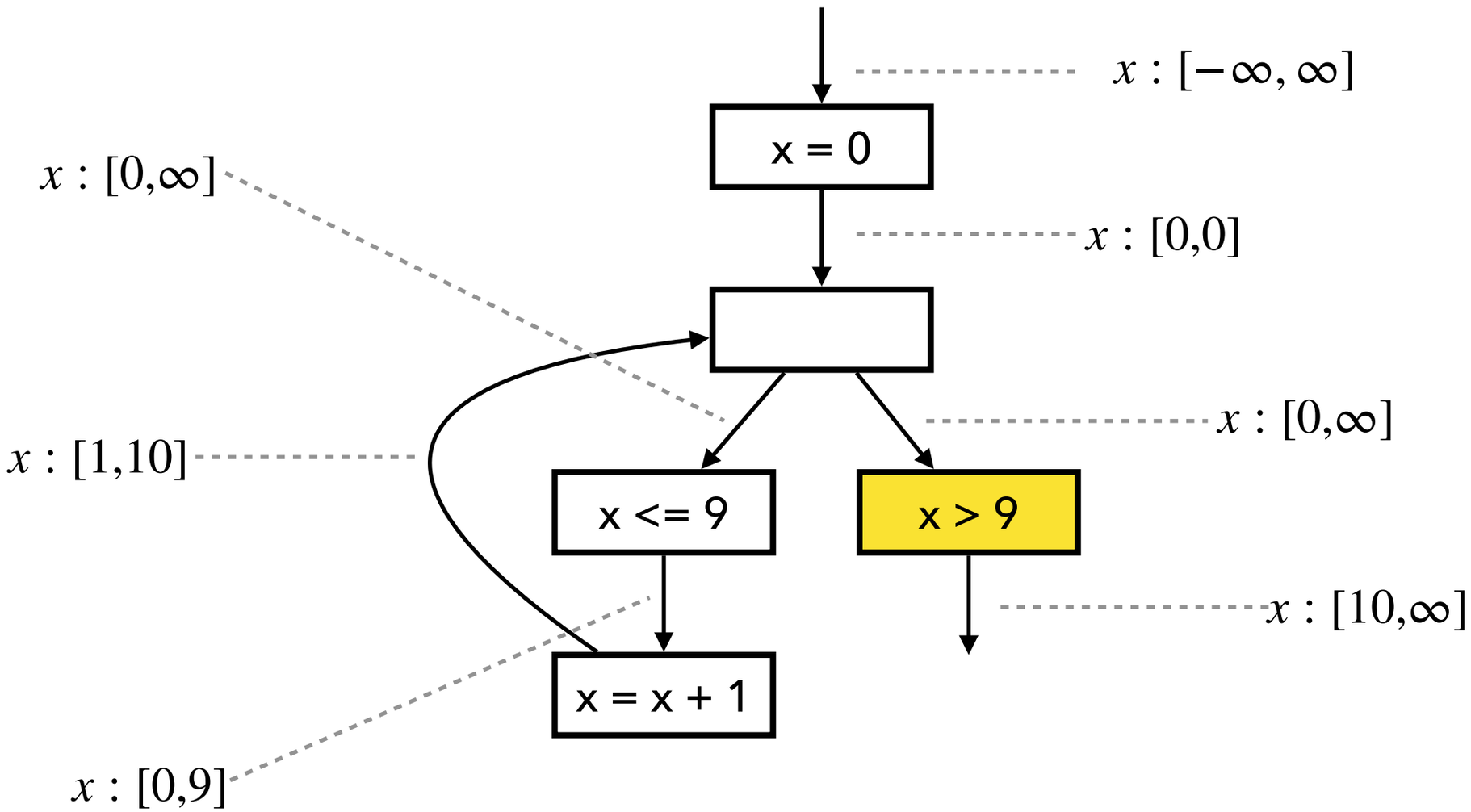
Fixed Point Comp. with Widening

3. Check if fixed point is reached

$$[0,\infty] \sqsupseteq [0,\infty]$$



Fixed Point Comp. with Widening

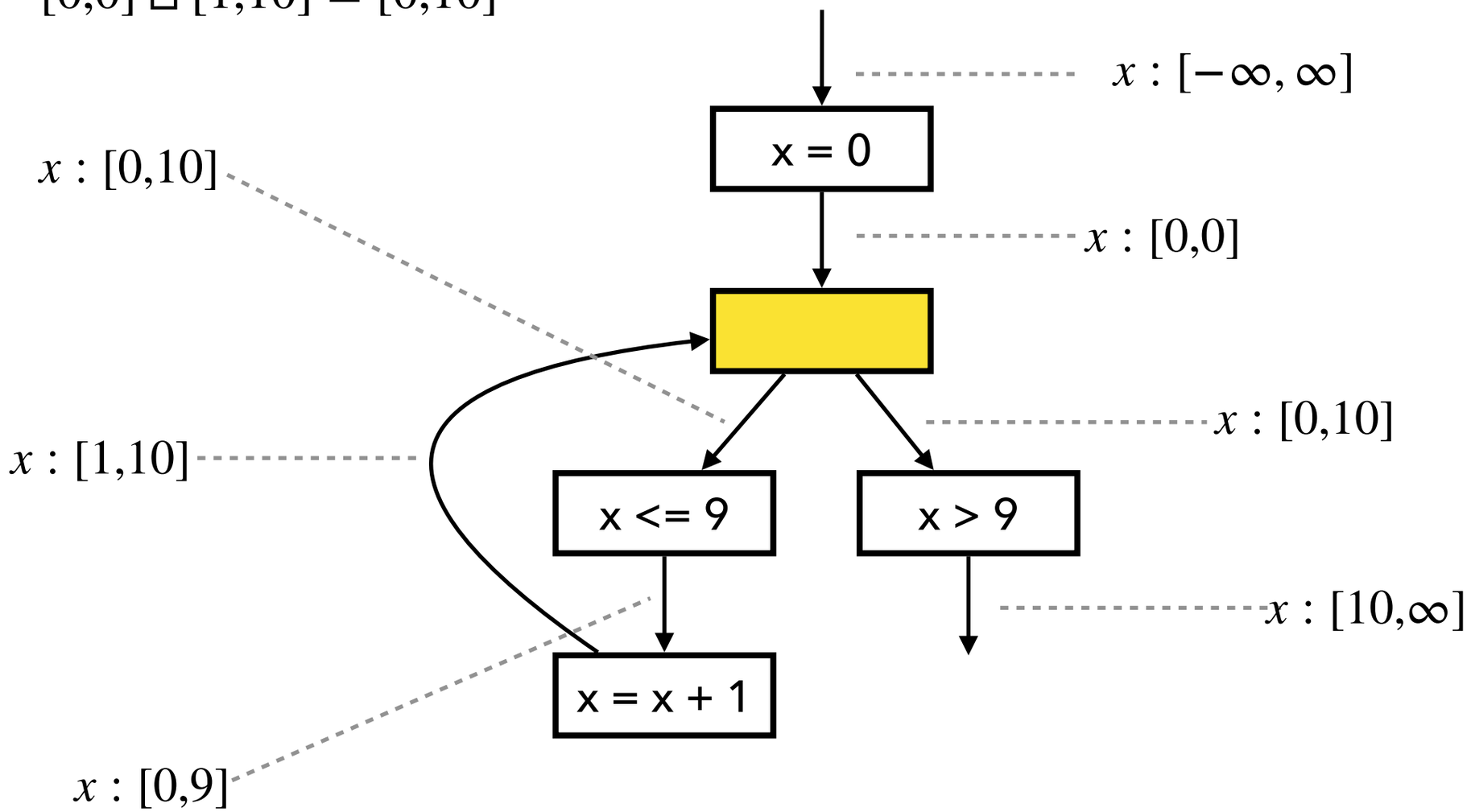


$$[0,\infty] \sqcap [10,\infty] = [10,\infty]$$

Fixed Point Comp. with Narrowing

1. Compute output by joining inputs:

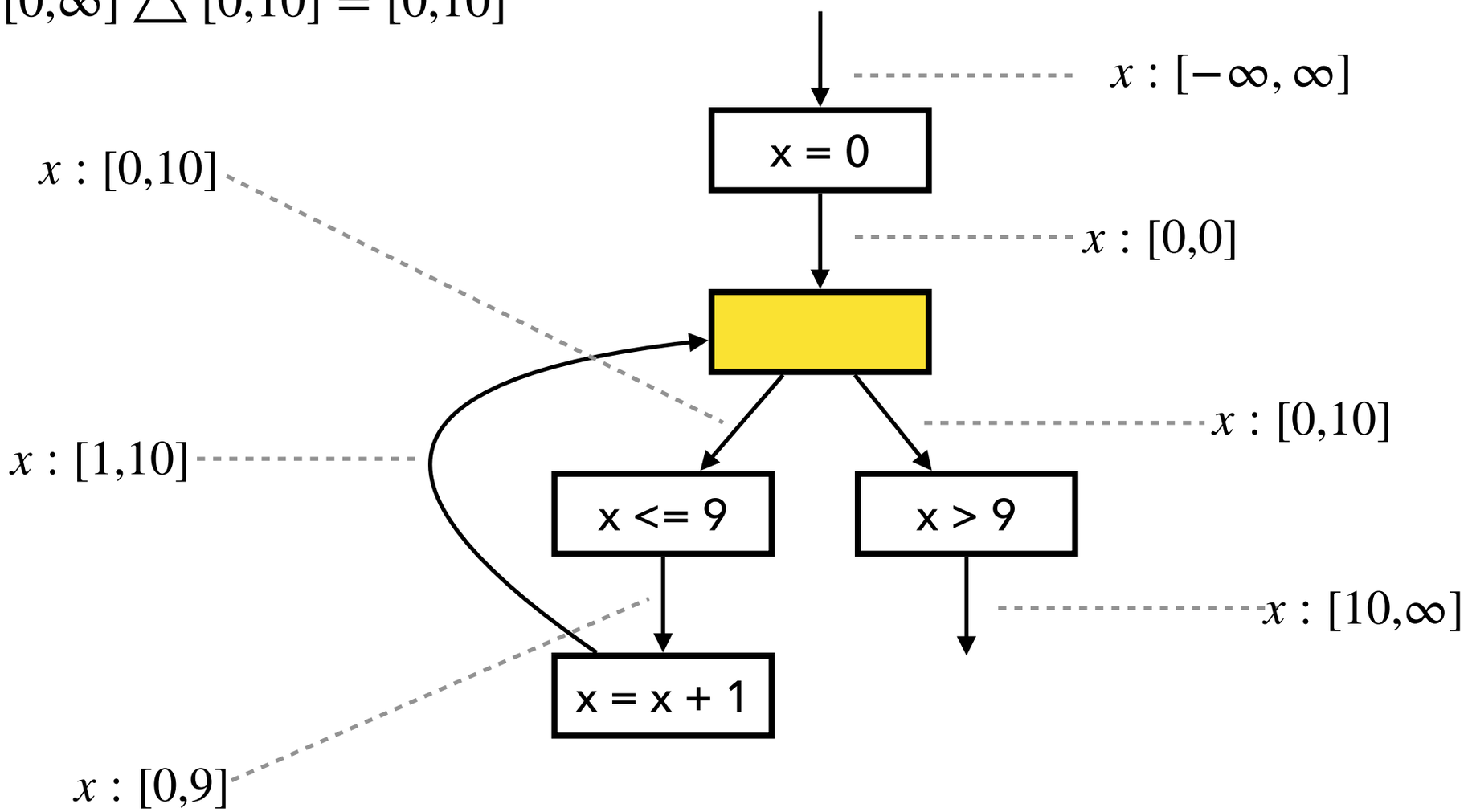
$$[0,0] \sqcup [1,10] = [0,10]$$



Fixed Point Comp. with Narrowing

2. Apply narrowing with old output:

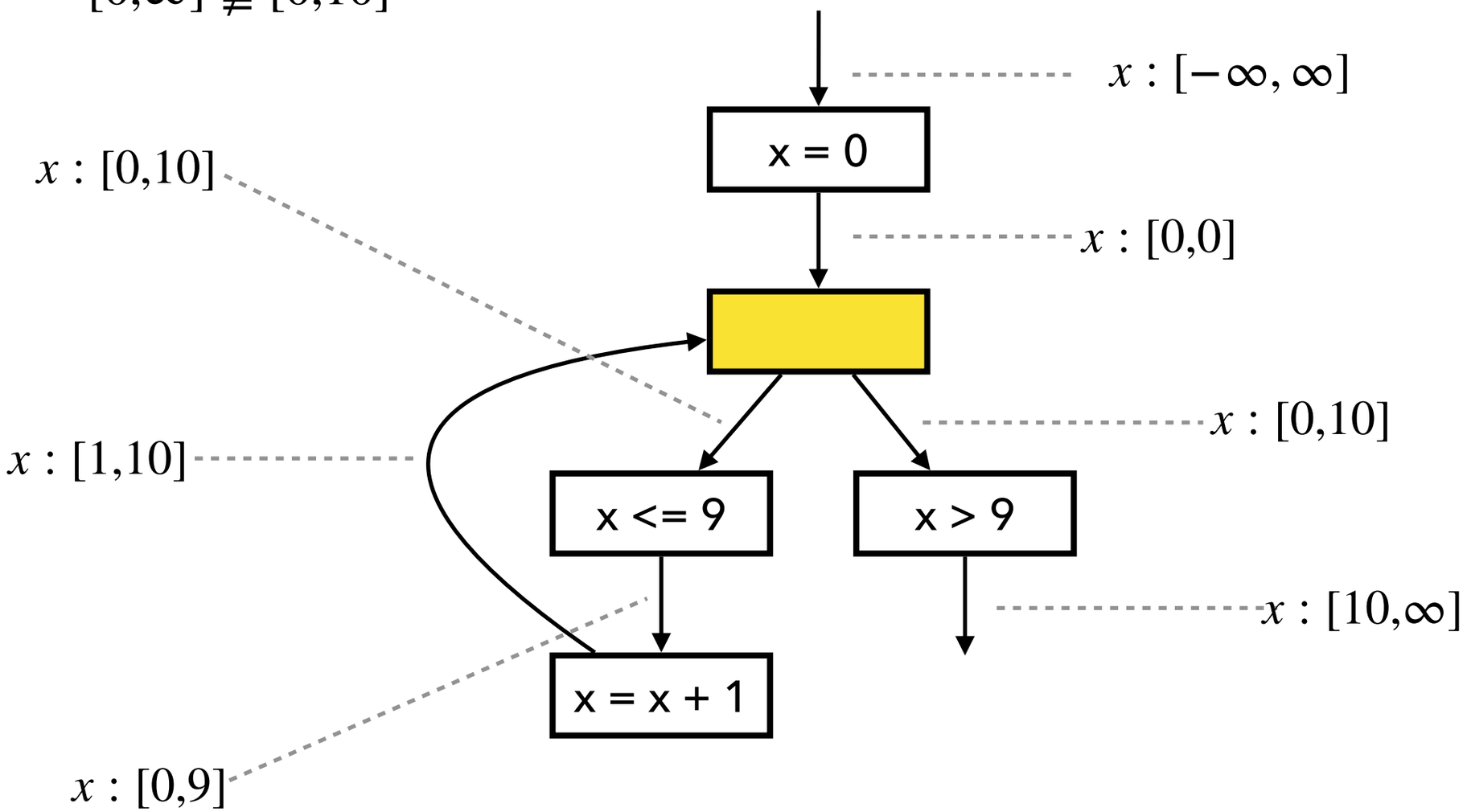
$$[0,\infty] \triangle [0,10] = [0,10]$$



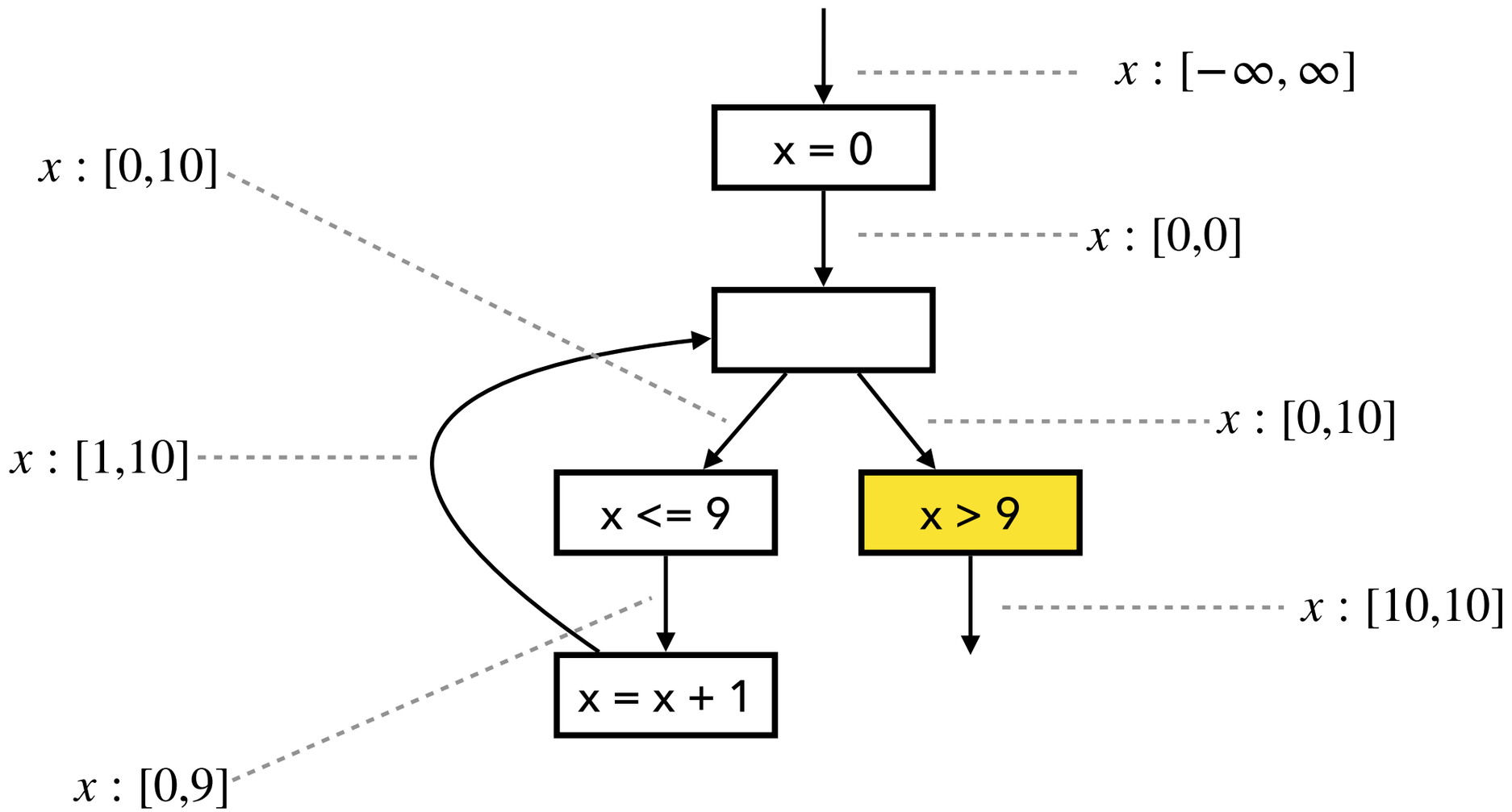
Fixed Point Comp. with Narrowing

3. Check if fixed point is reached:

$[0,\infty] \not\sqsubseteq [0,10]$



Fixed Point Comp. with Narrowing



The Interval Domain

- The set of intervals:

$$\hat{\mathbb{Z}} = \{ \bot \} \cup \{[l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, \infty\}, l \leq u\}$$

- Partial order:

$$\bot \sqsubseteq \hat{z} \quad \text{(for any } \hat{z} \in \hat{\mathbb{Z}}) \qquad [l_1, u_1] \sqsubseteq [l_2, u_2] \iff l_2 \leq l_1 \wedge u_1 \leq u_2$$

- Join:

$$\bot \sqcup \hat{z} = \hat{z} \qquad \hat{z} \sqcup \bot = \hat{z} \qquad [l_1, u_1] \sqcup [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$$

- Meet:

$$[l_1, u_1] \sqcap [l_2, u_2] = [l_2, u_1] \quad (\text{if } l_1 \leq l_2 \wedge l_2 \leq u_1)$$

$$[l_1, u_1] \sqcap [l_2, u_2] = [l_1, u_2] \quad (\text{if } l_2 \leq l_1 \wedge l_1 \leq u_2)$$

$$\hat{z}_1 \sqcap \hat{z}_2 = \bot \quad \text{(otherwise)}$$

The Interval Domain

- Widening:

$$\bot \bigtriangledown \hat{z} = \hat{z}$$
$$\hat{z} \bigtriangledown \bot = \hat{z}$$
$$[l_1, u_1] \bigtriangledown [l_2, u_2] = [l_1 > l_2 ? -\infty : l_1, u_1 < u_2 ? +\infty : u_1]$$

- Narrowing:

$$\bot \bigtriangleup \hat{z} = \bot$$
$$\hat{z} \bigtriangleup \bot = \bot$$
$$[l_1, u_1] \bigtriangleup [l_2, u_2] = [l_1 = -\infty ? l_2 : l_1, u_1 = +\infty ? u_2 : u_1]$$

The Interval Domain

- Addition / Subtraction / Multiplication:

$$[l_1, u_1] \hat{+} [l_2, u_2] = [l_1 + l_2, u_1 + u_2]$$

$$[l_1, u_1] \hat{-} [l_2, u_2] = [l_1 - u_2, u_1 - l_2]$$

$$[l_1, u_1] \hat{\times} [l_2, u_2] = [\min(l_1 l_2, l_1 u_2, u_1 l_2, u_1 u_2), \max(l_1 l_2, l_1 u_2, u_1 l_2, u_1 u_2)]$$

- Equality (=) produces T except for the cases:

$$[l_1, u_1] \hat{=} [l_2, u_2] = \hat{true} \quad (\text{if } l_1 = u_1 = l_2 = u_2)$$

$$[l_1, u_1] \hat{=} [l_2, u_2] = \hat{false} \quad (\text{no overlap})$$

- ``Less than'' (<) produces T except for the cases:

$$[l_1, u_1] \hat{<} [l_2, u_2] = \hat{true} \quad (\text{if } u_1 < l_2)$$

$$[l_1, u_1] \hat{<} [l_2, u_2] = \hat{false} \quad (\text{if } l_1 > u_2)$$

Abstract Memory

$$\hat{\mathbb{M}} = \textbf{Var} \to \hat{\mathbb{Z}}$$

- Pointwise extension of ordering, join, widening, narrowing:

$$m_1 \sqsubseteq m_2 \iff \forall x \in \textbf{Var} \,.\, m_1(x) \sqsubseteq m_2(x)$$

$$m_1 \sqcup m_2 = \lambda x \,.\, m_1(x) \sqcup m_2(x)$$

$$m_1 \bigtriangledown m_2 = \lambda x \,.\, m_1(x) \bigtriangledown m_2(x)$$

$$m_1 \bigtriangleup m_2 = \lambda x \,.\, m_1(x) \bigtriangleup m_2(x)$$

Worklist Algorithm

Fixpoint comp. with widening

$W := \textbf{Node}$
$T := \lambda n\,.\,\bot_{\hat{\mathbb{M}}}$
$while\ W \neq \emptyset$
  $n := choose(W)$
  $W := W \backslash \{n\}$
  $in := inputof(n, T)$
  $out := analyze(n, in)$
  $if\ out \not\sqsubseteq T(n)$
    $if\ widening\ is\ needed$
      $T(n) := T(n) \bigtriangledown out$
    $else$
      $T(n) := T(n) \sqcup out$
    $W := W \cup succ(n)$

Fixpoint comp. with narrowing

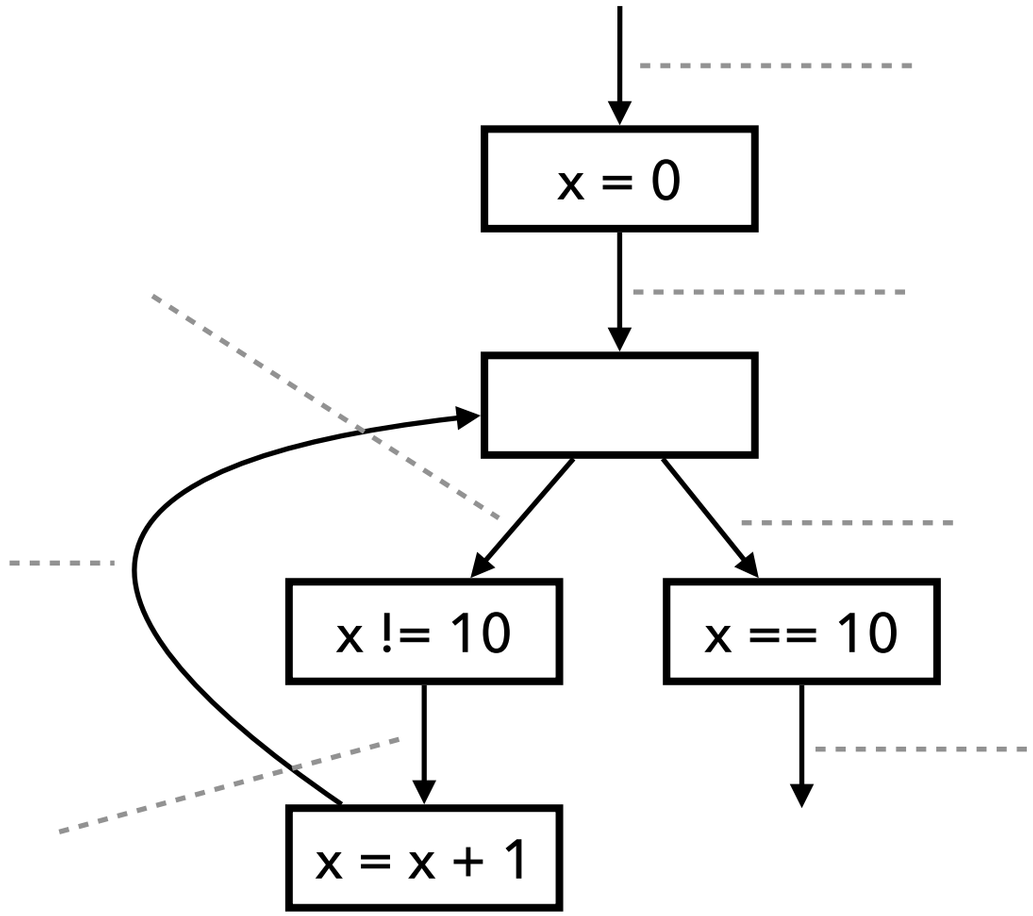
$W := \textbf{Node}$
$while\ W \neq \emptyset$
  $n := choose(W)$
  $W := W \backslash \{n\}$
  $in := inputof(n, T)$
  $out := analyze(n, in)$
  $if\ T(n) \not\sqsupseteq out$
    $T(n) := T(n) \bigtriangleup out$
    $W := W \cup succ(n)$

Exercise (2)

Describe the result of the interval analysis:
(1) without widening
(2) with widening/narrowing

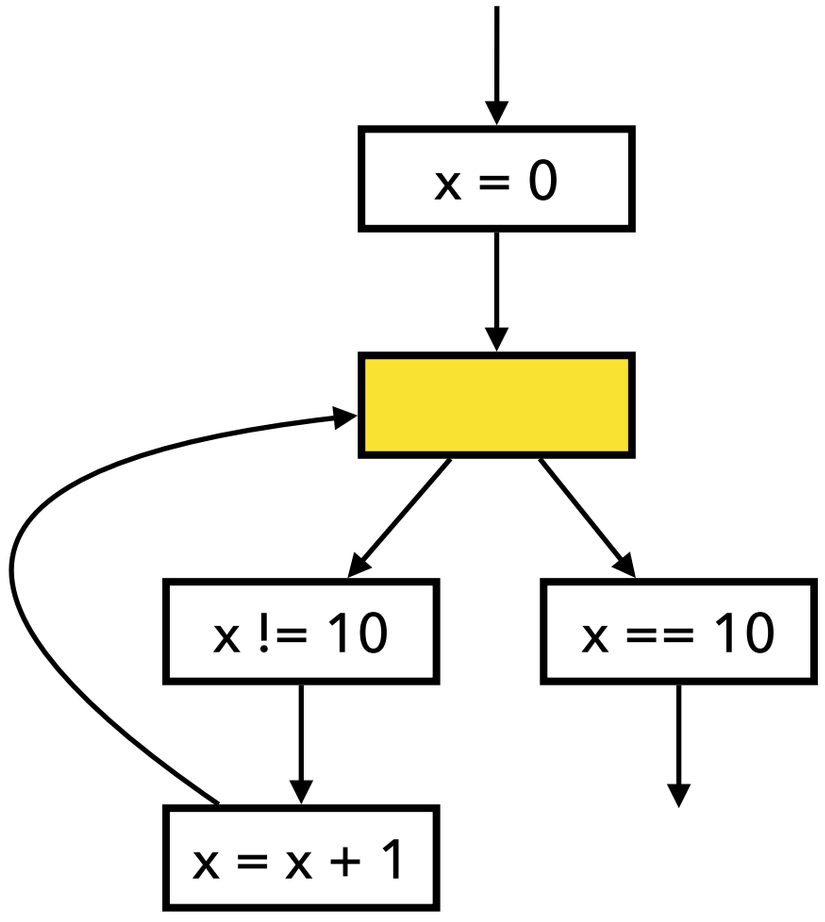
```
x = 0;

while (x != 10)
  x = x + 1;
```



Widening with Thresholds

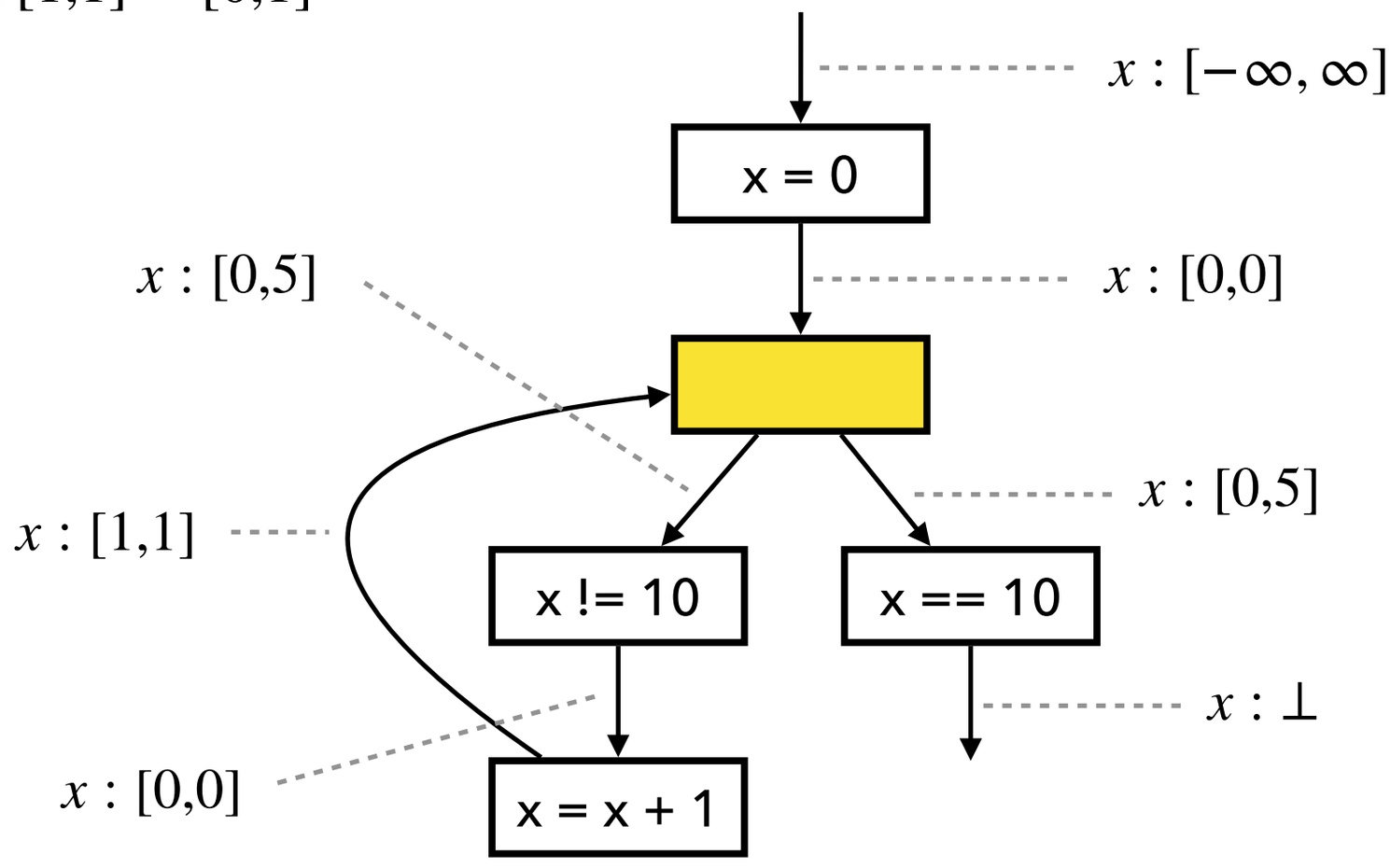
Assume a set T of thresholds is given beforehand: e.g., $T = \{5,10\}$



Widening with Thresholds

1. Compute output by joining inputs:

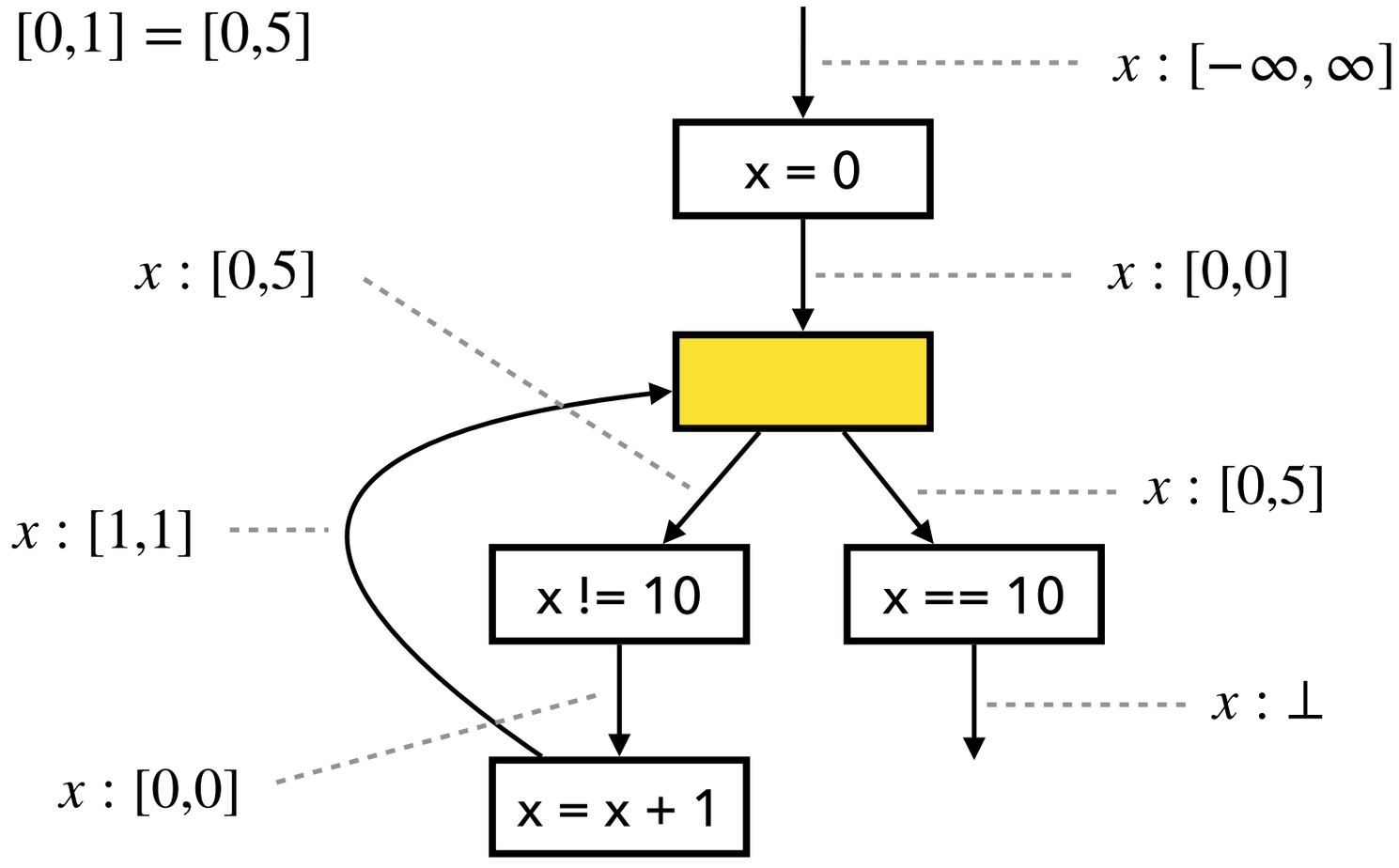
$$[0,0] \sqcup [1,1] = [0,1]$$



Widening with Thresholds

2. Given $T = \{5,10\}$, use 5 as threshold when applying widening:

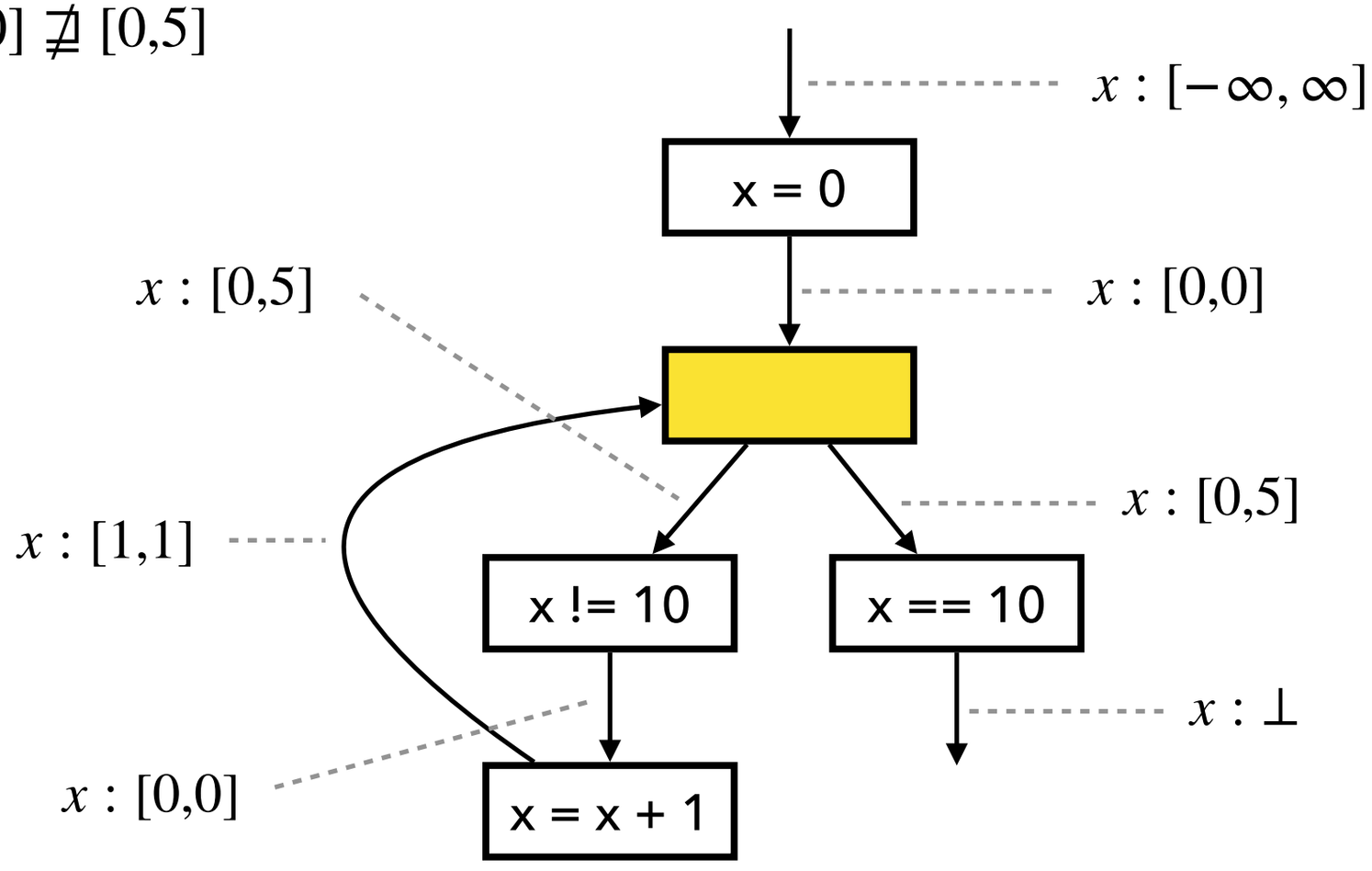
$$[0,0] \triangledown [0,1] = [0,5]$$



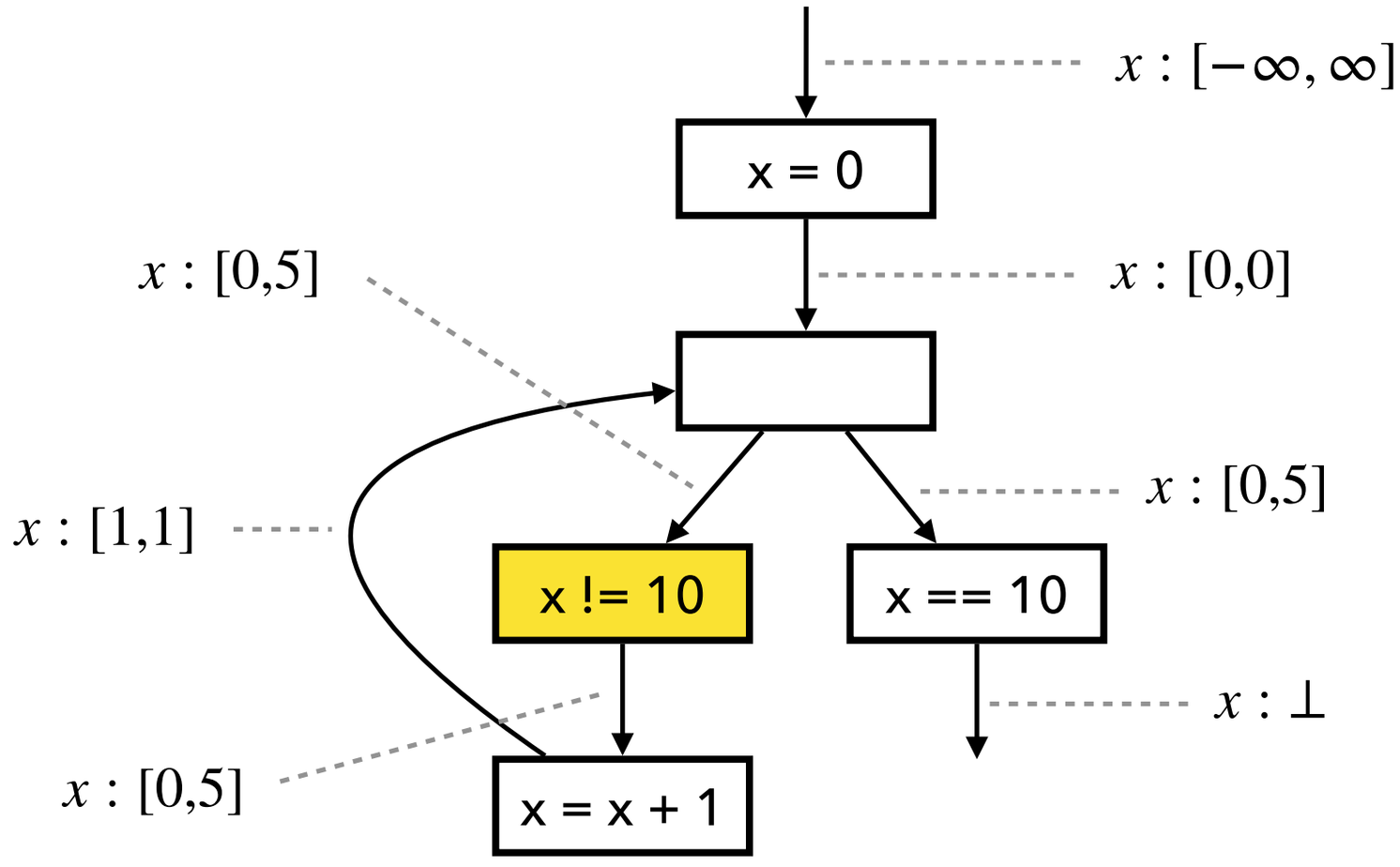
Widening with Thresholds

3. Check if fixed point is reached:

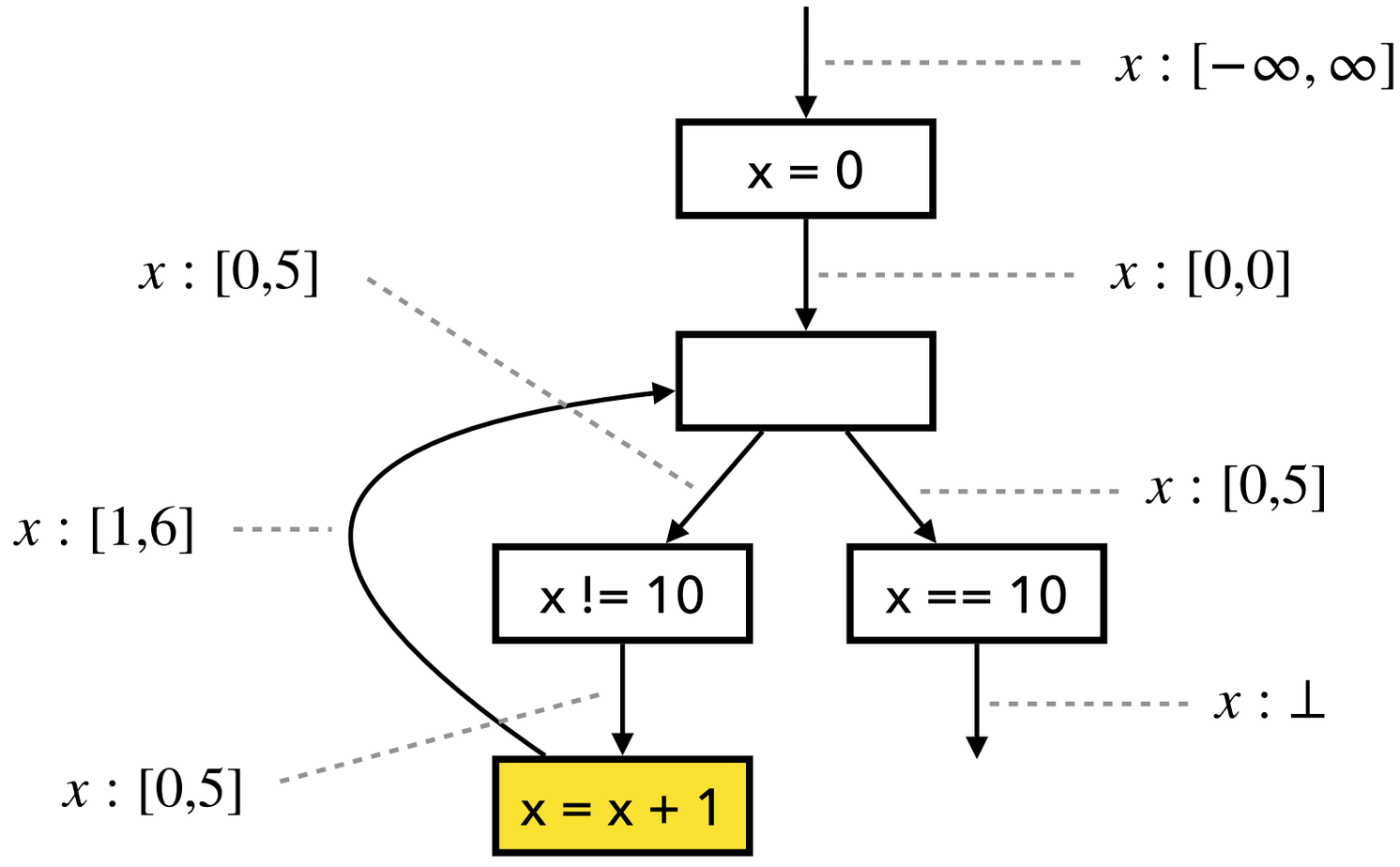
$$[0,0] \not\supseteq [0,5]$$



Widening with Thresholds



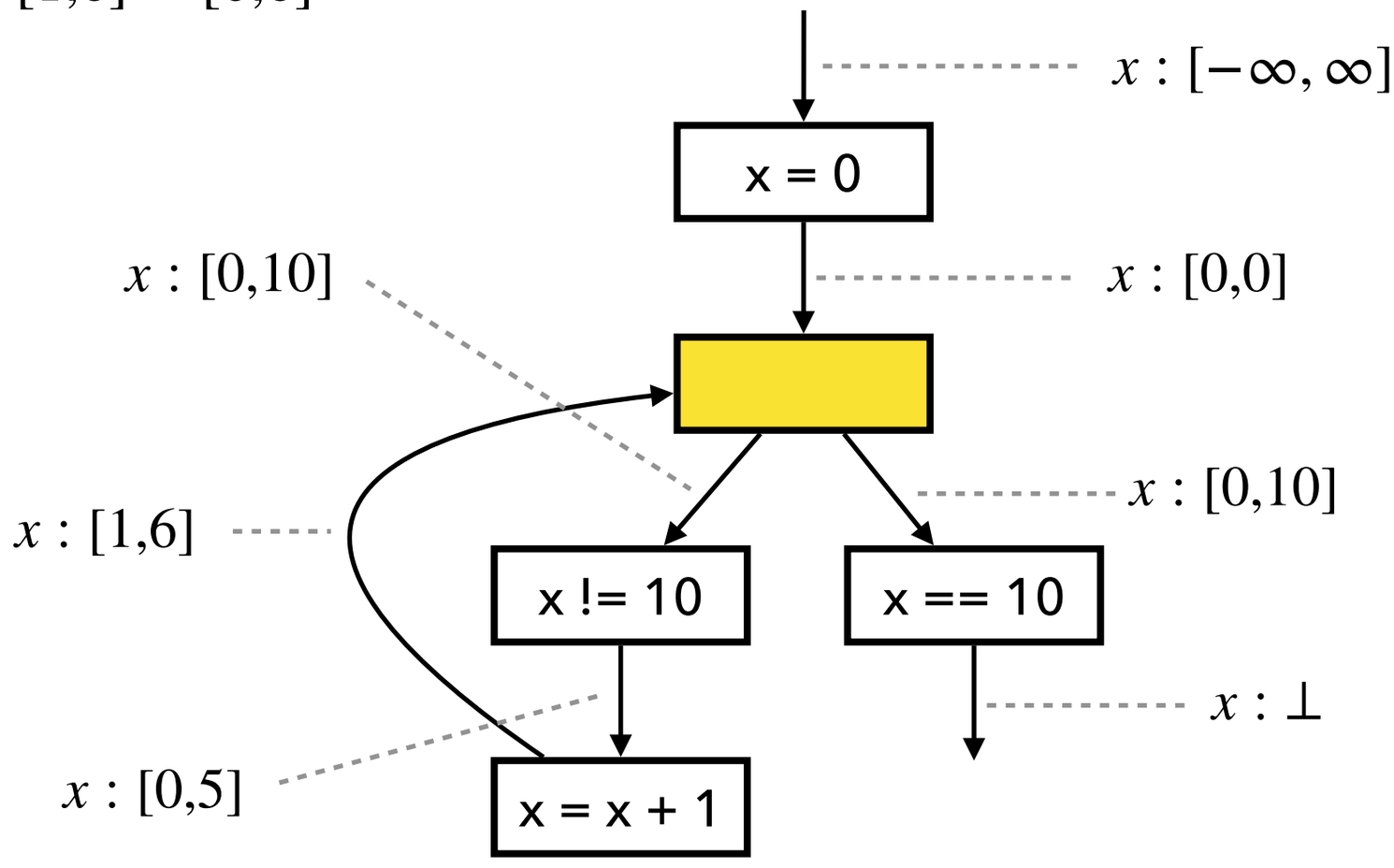
Widening with Thresholds



Widening with Thresholds

1. Compute output by joining inputs:

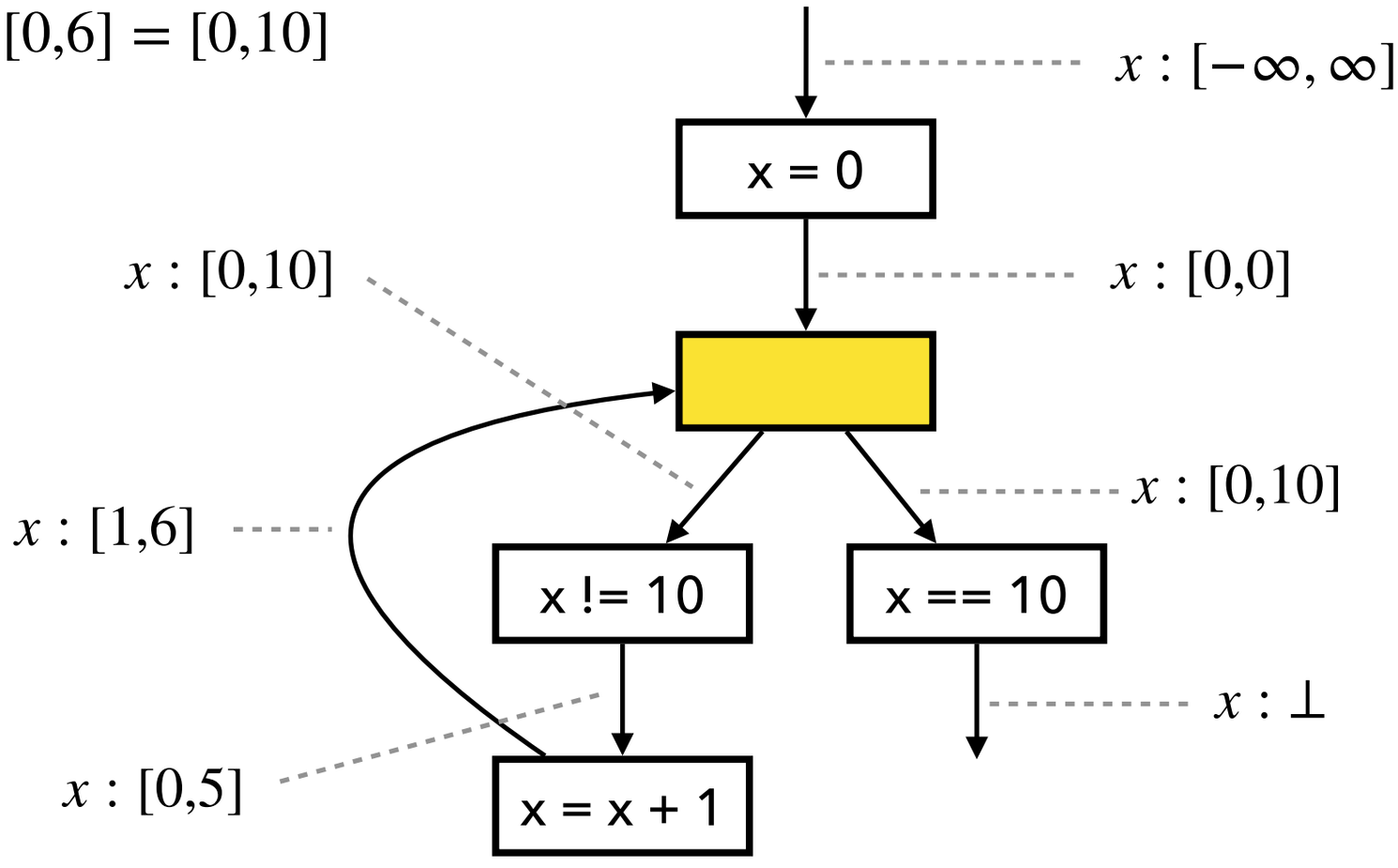
$$[0,0] \sqcup [1,6] = [0,6]$$



Widening with Thresholds

2. Given $T = \{5,10\}$, use 10 as threshold when applying widening:

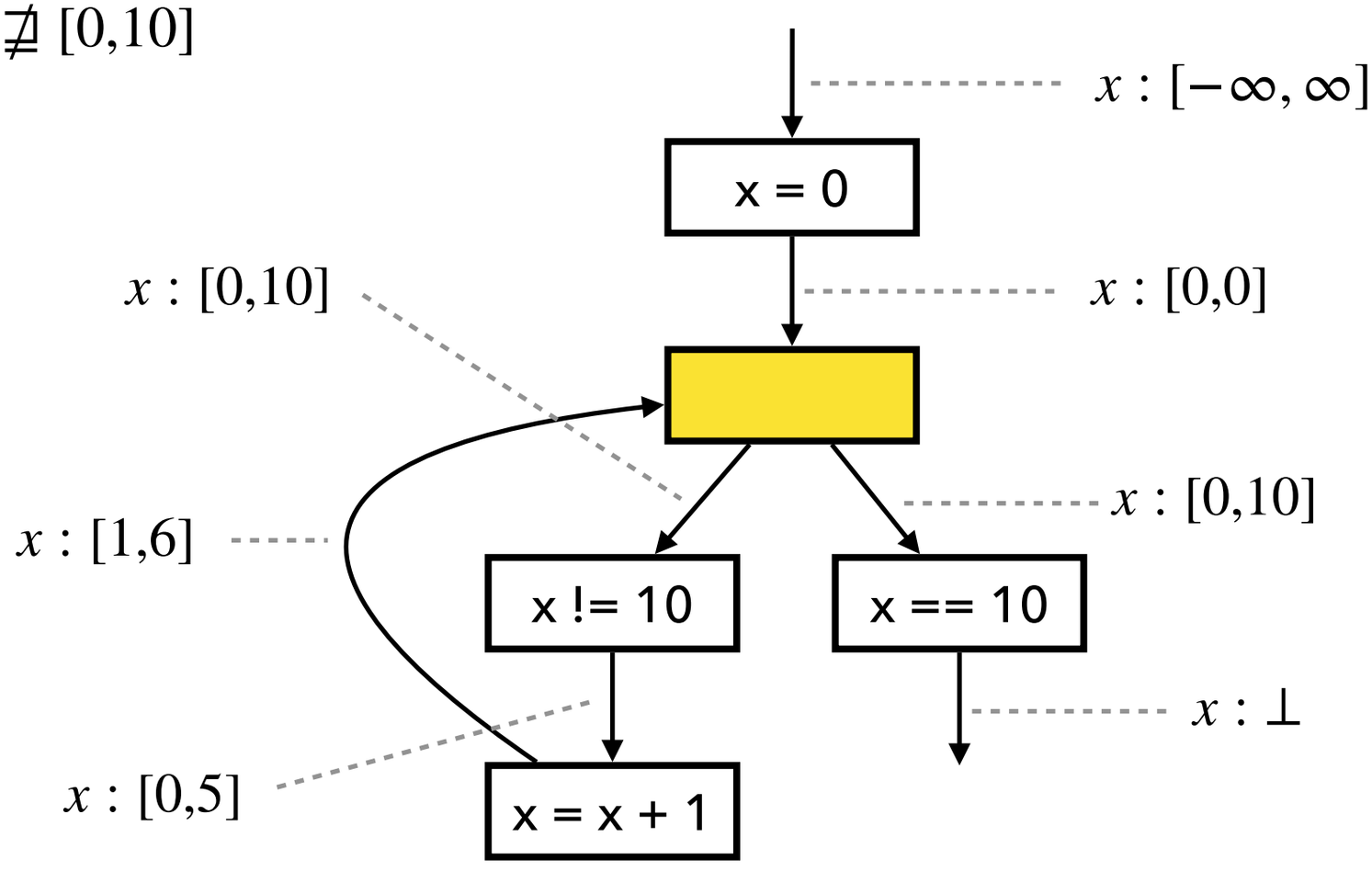
$$[0,5] \triangledown [0,6] = [0,10]$$



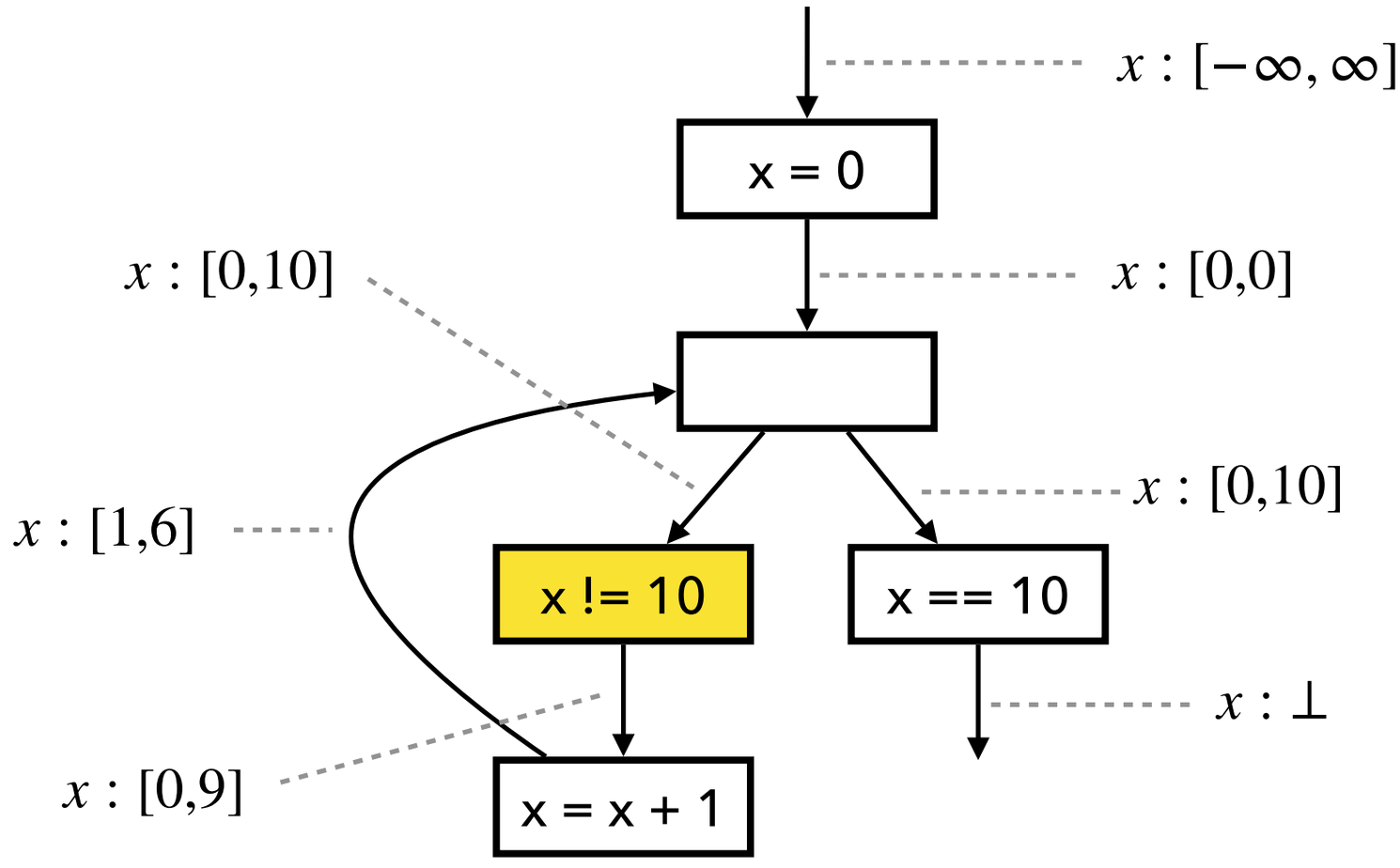
Widening with Thresholds

3. Check if fixed point is reached:

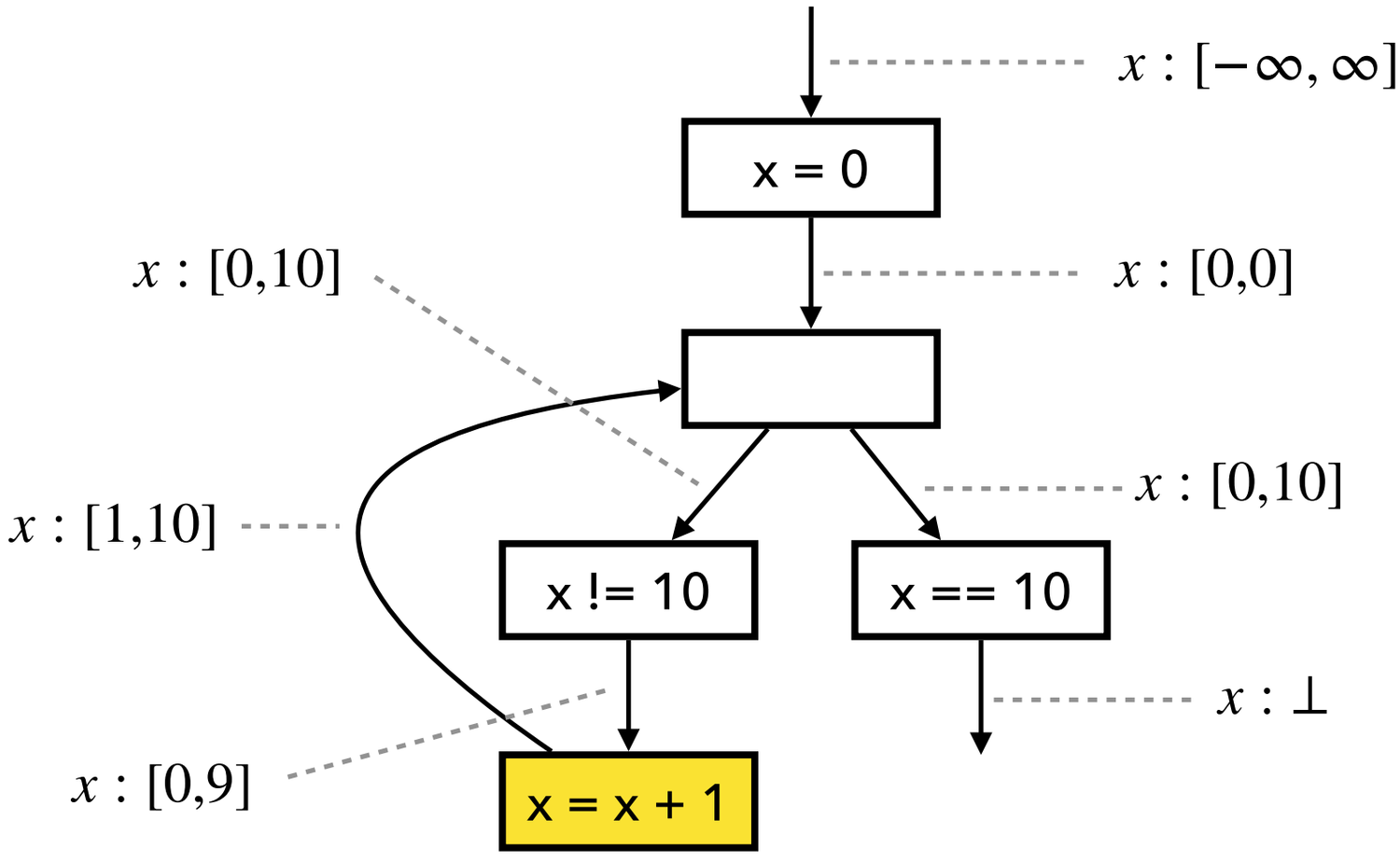
$$[0,5] \not\supseteq [0,10]$$



Widening with Thresholds



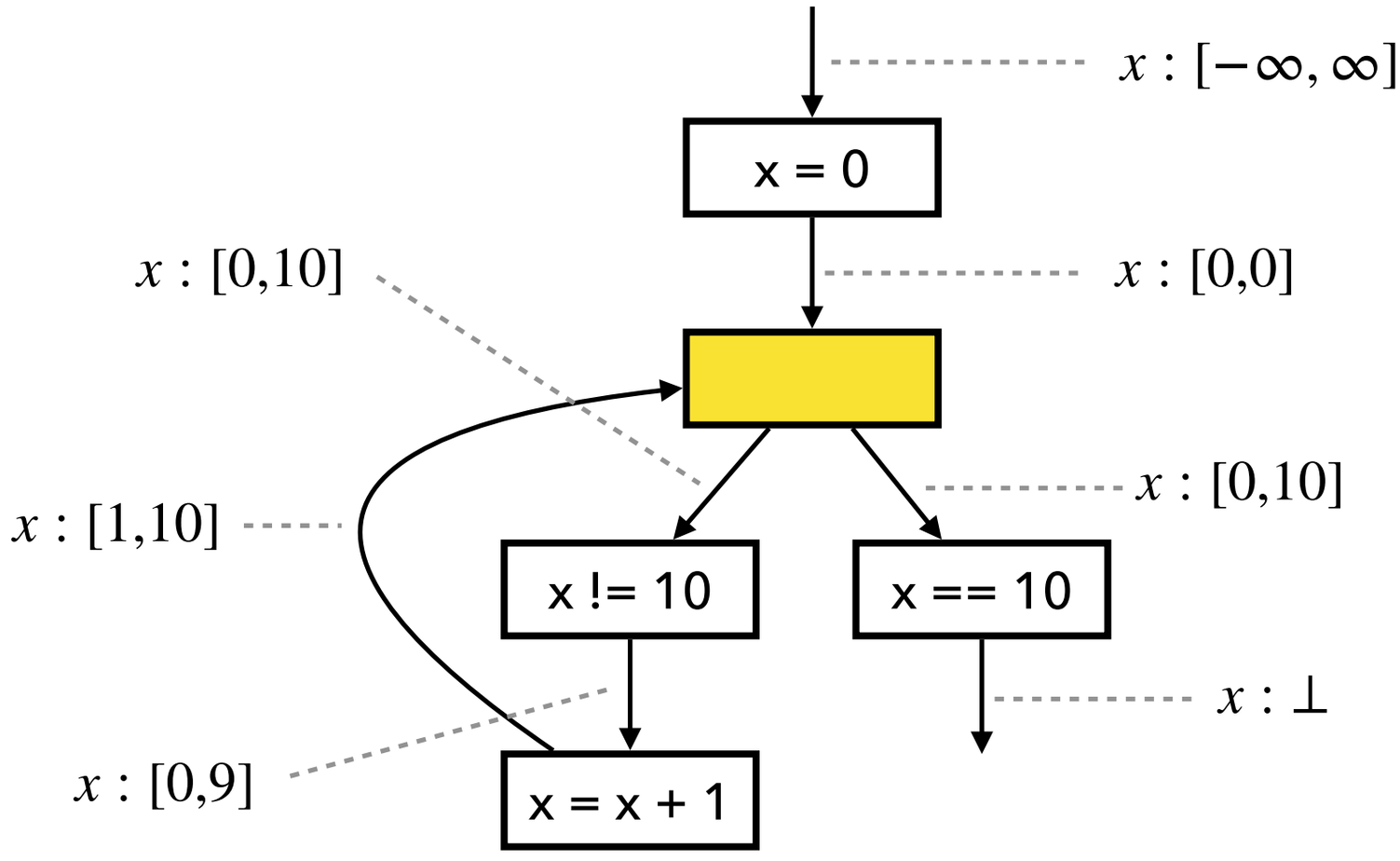
Widening with Thresholds



Widening with Thresholds

1. Compute output by joining inputs:

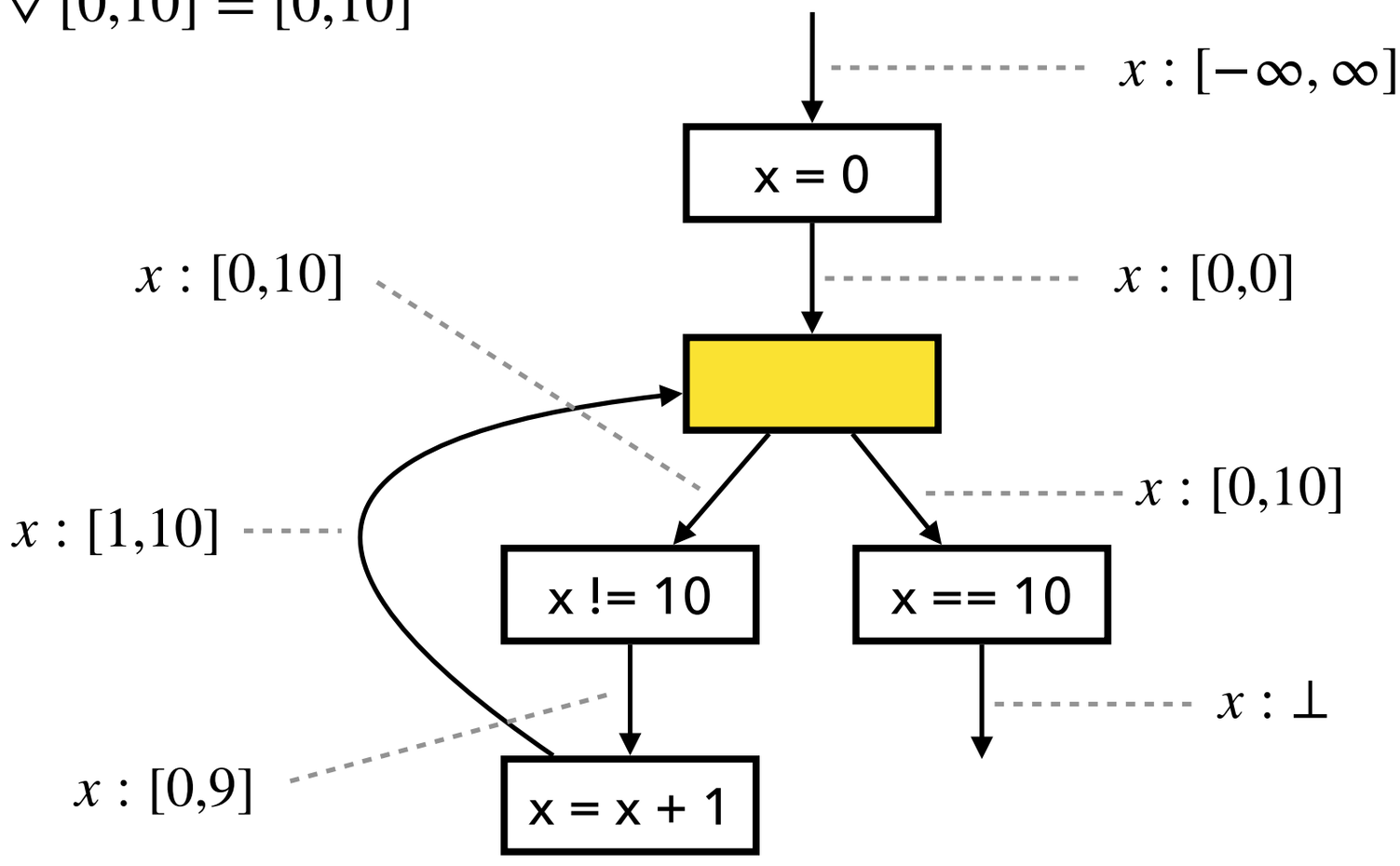
$$[0,0] \sqcup [1,10] = [0,10]$$



Widening with Thresholds

2. Apply widening:

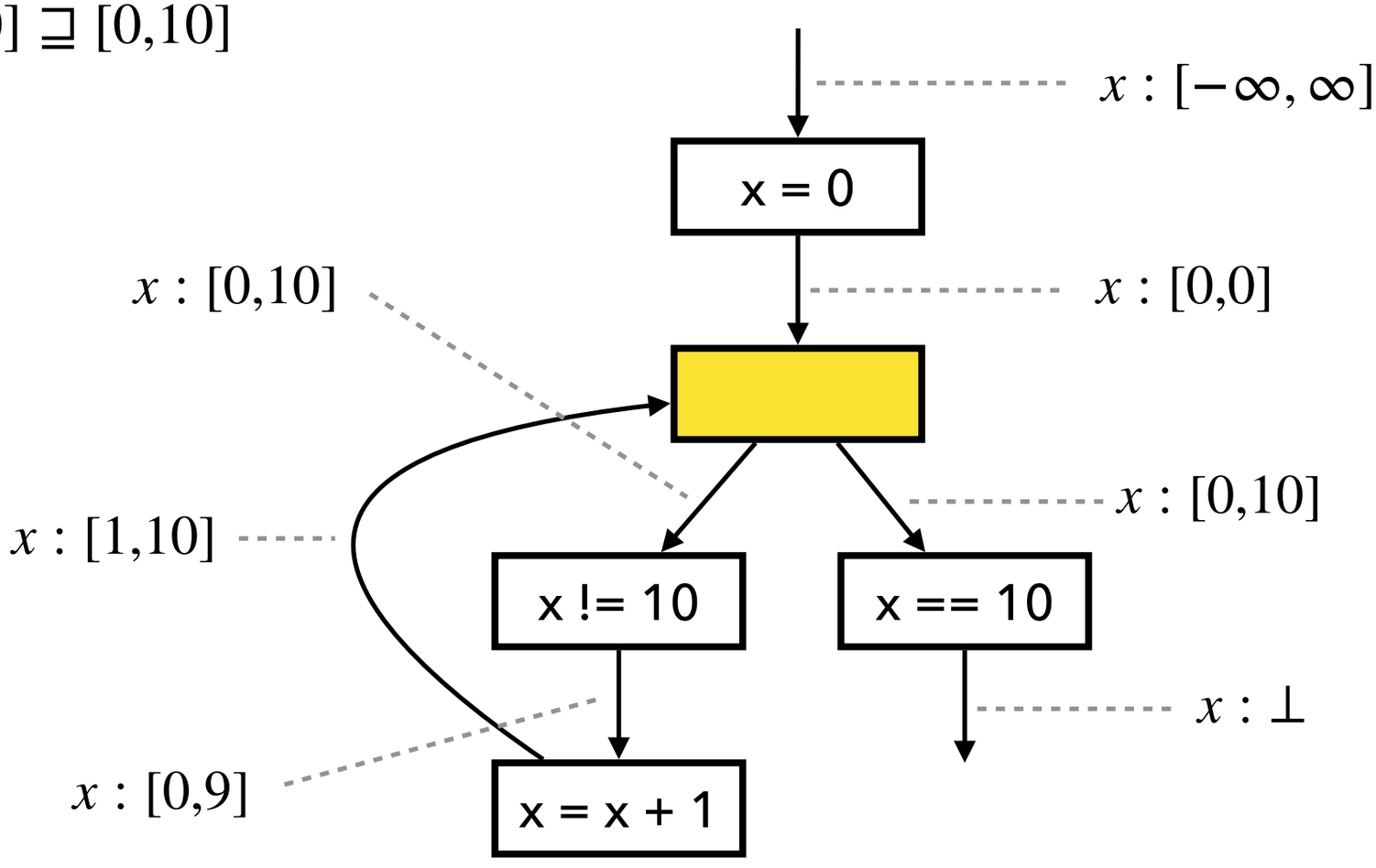
$$[0,10] \triangledown [0,10] = [0,10]$$



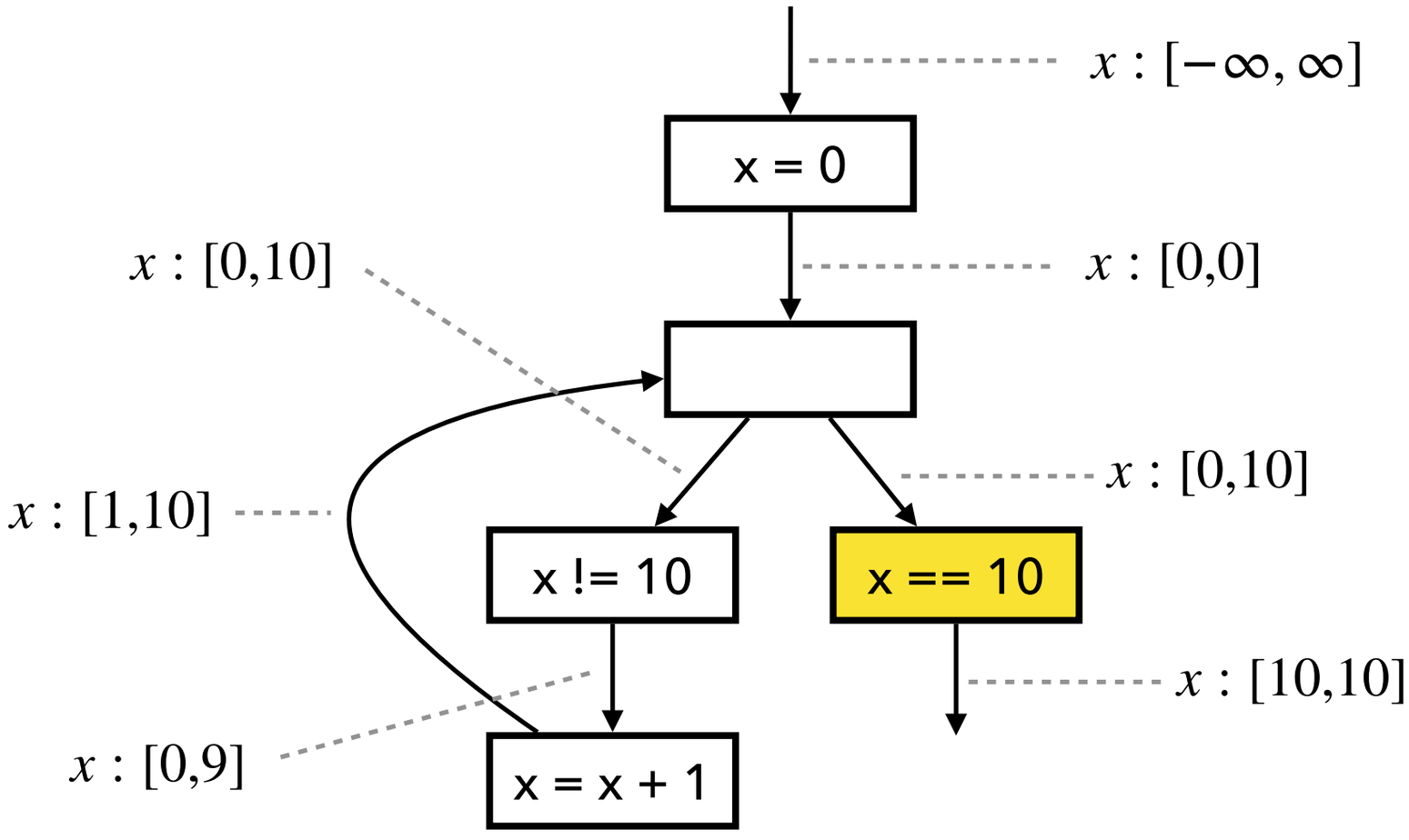
Widening with Thresholds

3. Check if fixed point is reached:

$$[0,10] \sqsupseteq [0,10]$$



Widening with Thresholds



Widening with Thresholds

- A threshold set $T \subseteq \mathbb{Z}$ is given.

$$\bot \bigtriangledown_T \hat{z} = \hat{z}$$
$$\hat{z} \bigtriangledown_T \bot = \hat{z}$$
$$[l_1, u_1] \bigtriangledown_T [l_2, u_2] = [l_1 > l_2 ? glb(T, l_2) : l_1, u_1 < u_2 ? lub(T, u_2) : u_1]$$

$$glb(T, n) = max\{t \in T \mid t \leq n\}$$
$$lub(T, n) = min\{t \in T \mid t \geq n\}$$

Exercise (3)

Describe the result of the interval analysis with widening and narrowing

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