

# Program Analysis

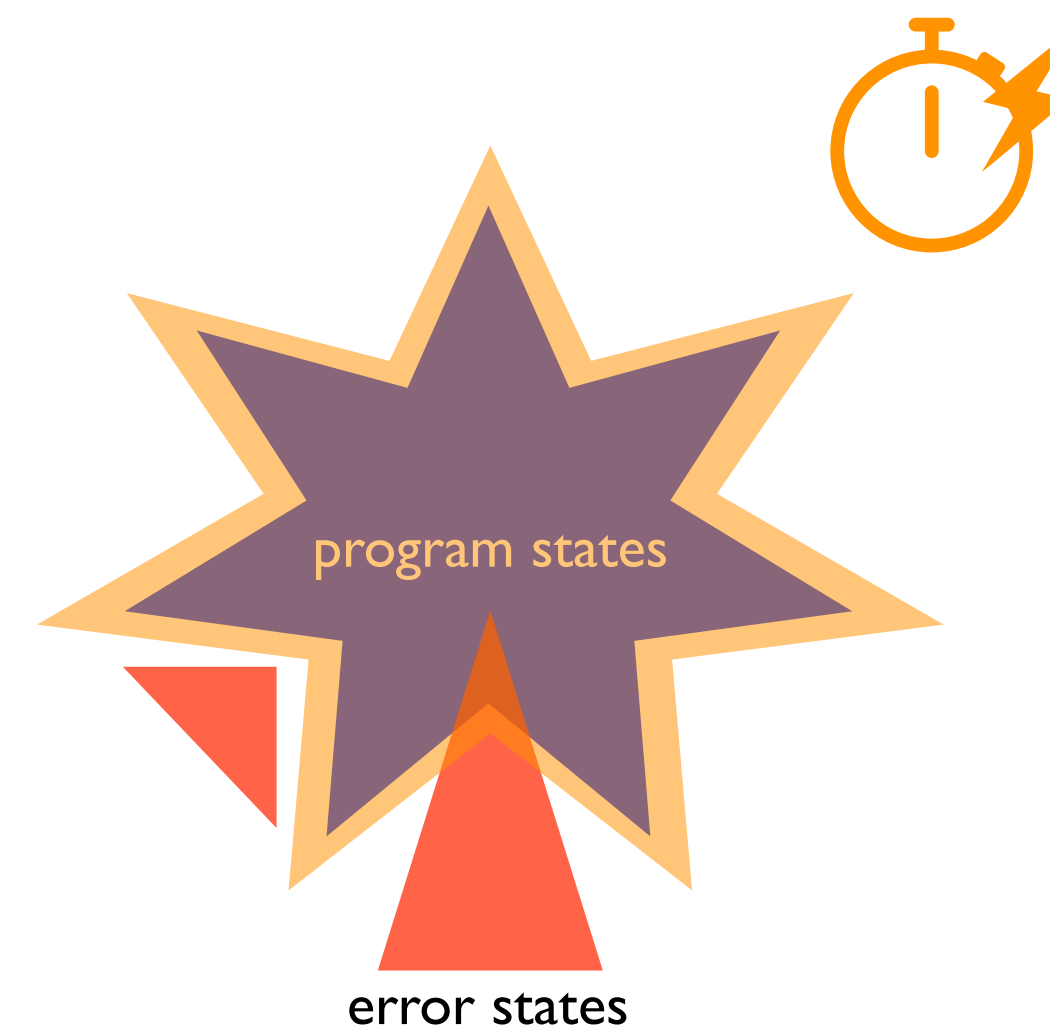
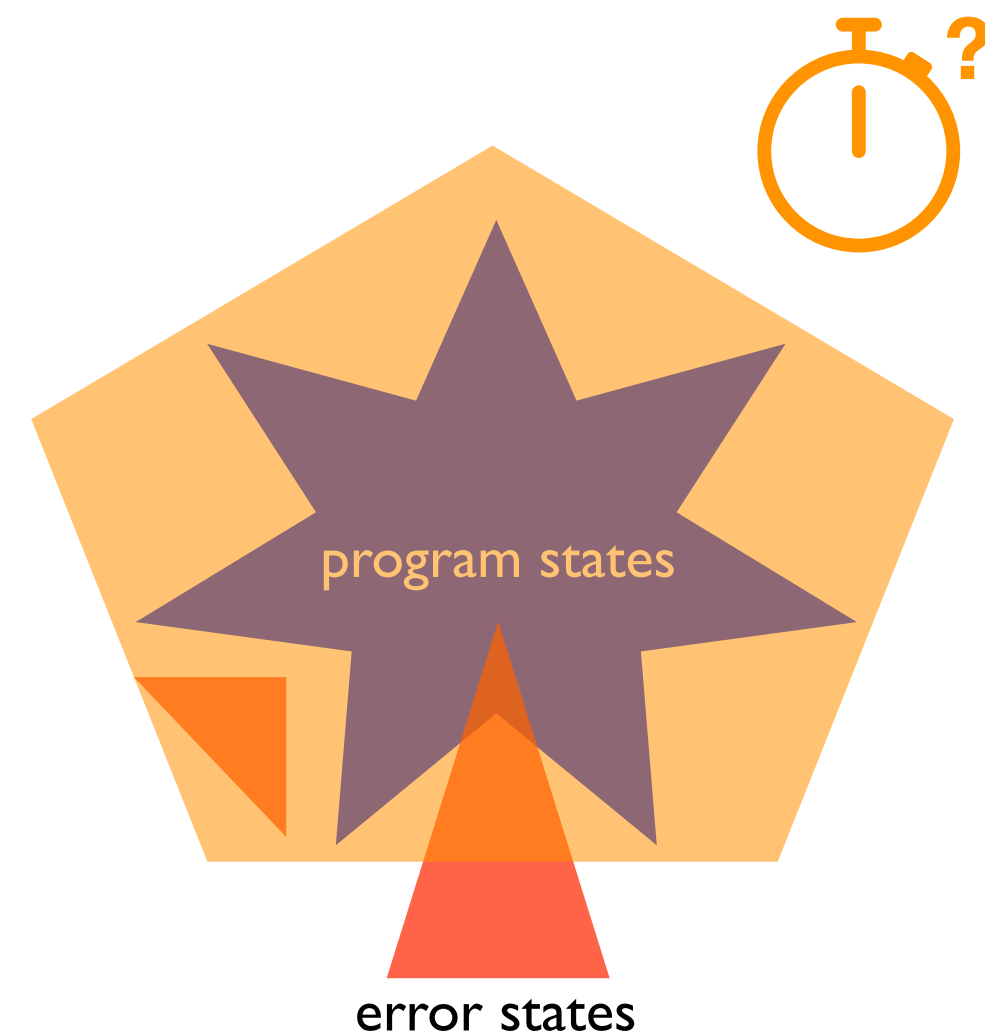
## 10. Advanced Iteration Techniques

Kihong Heo



# Advanced Analysis Techniques

- So far, our focus most has been **sound** abstract semantics
- From now on, we will cover several advanced techniques to achieve **efficient** and **accurate** analysis



# Iteration Strategies

- **Loop invariant inference**: sequences of abstract iterations
  - Compute **weaker** and **weaker** abstract states until stabilization (via join and widening)
- “***Loop is evil***”: a main source of imprecision in static analysis
- Needs for techniques to improve the precision

# Problem 1: Overused Widening

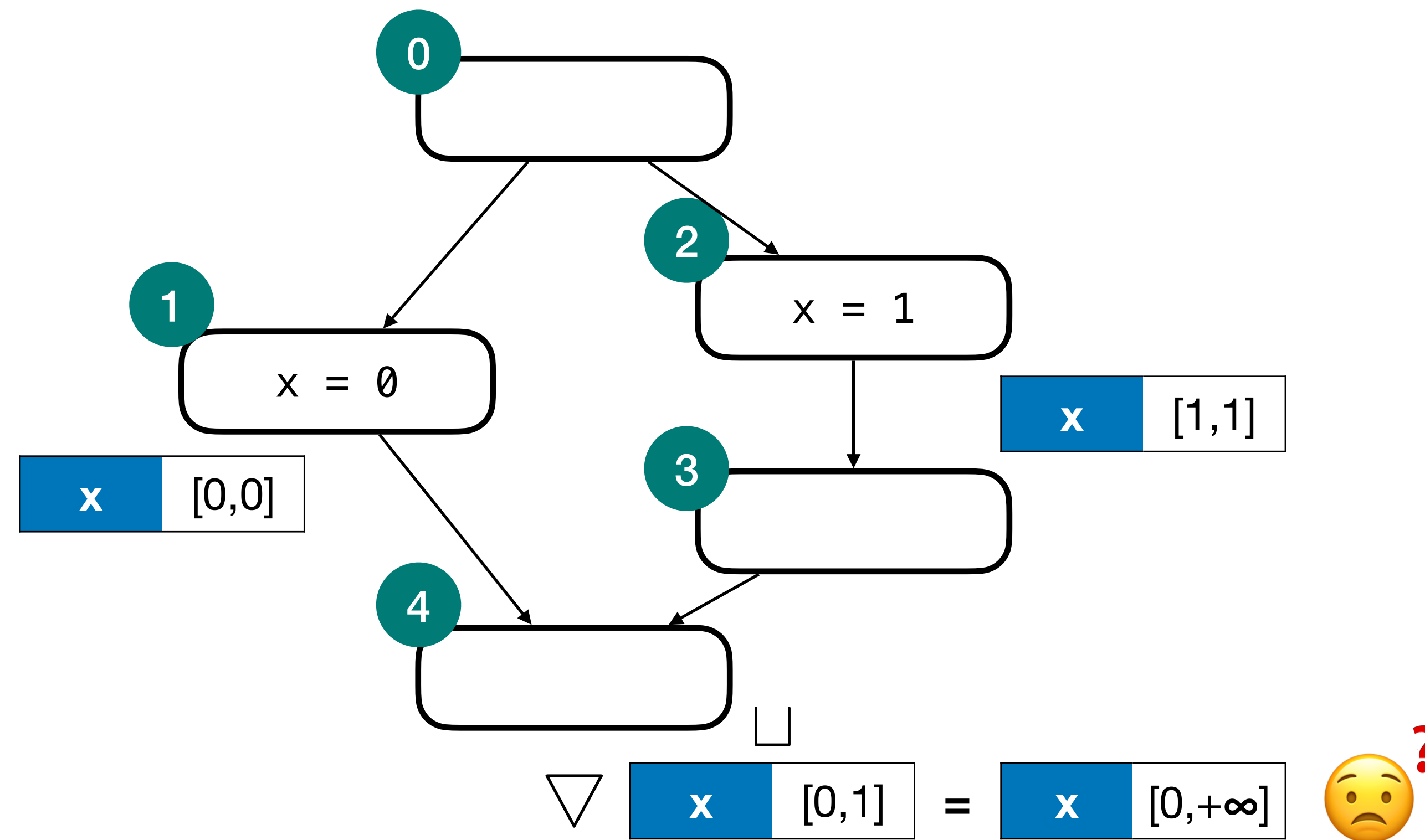
- Recall the worklist algorithm

```
 $X : \mathbb{L} \rightarrow \mathbb{M}^\sharp$   
 $F^\sharp : (\mathbb{L} \rightarrow \mathbb{M}^\sharp) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}^\sharp)$   
 $Worklist : \wp(\mathbb{L})$   
begin  
   $Worklist \leftarrow \mathbb{L}$   
   $X \leftarrow \perp$   
  repeat  
     $(w, Worklist) \leftarrow pop$   
     $m_{old}^\sharp \leftarrow X(w)$   
     $m_{new}^\sharp \leftarrow \bigsqcup \{m_{in}^\sharp \mid \langle w, m_{in}^\sharp \rangle \in F^\sharp(X)\}$   
     $m_{new}^\sharp \leftarrow m_{old}^\sharp \nabla m_{new}^\sharp$   
    if  $m_{new}^\sharp \not\sqsubseteq m_{old}^\sharp$  then  
       $X(w) \leftarrow m_{new}^\sharp$   
       $Worklist \leftarrow Worklist \cup \{l \mid \langle w, m_{new}^\sharp \rangle \hookrightarrow^\sharp \langle l, - \rangle\}$   
    endif  
  until  $Worklist = \emptyset$   
  return  $X$   
end
```

Widening Everywhere?

# Example

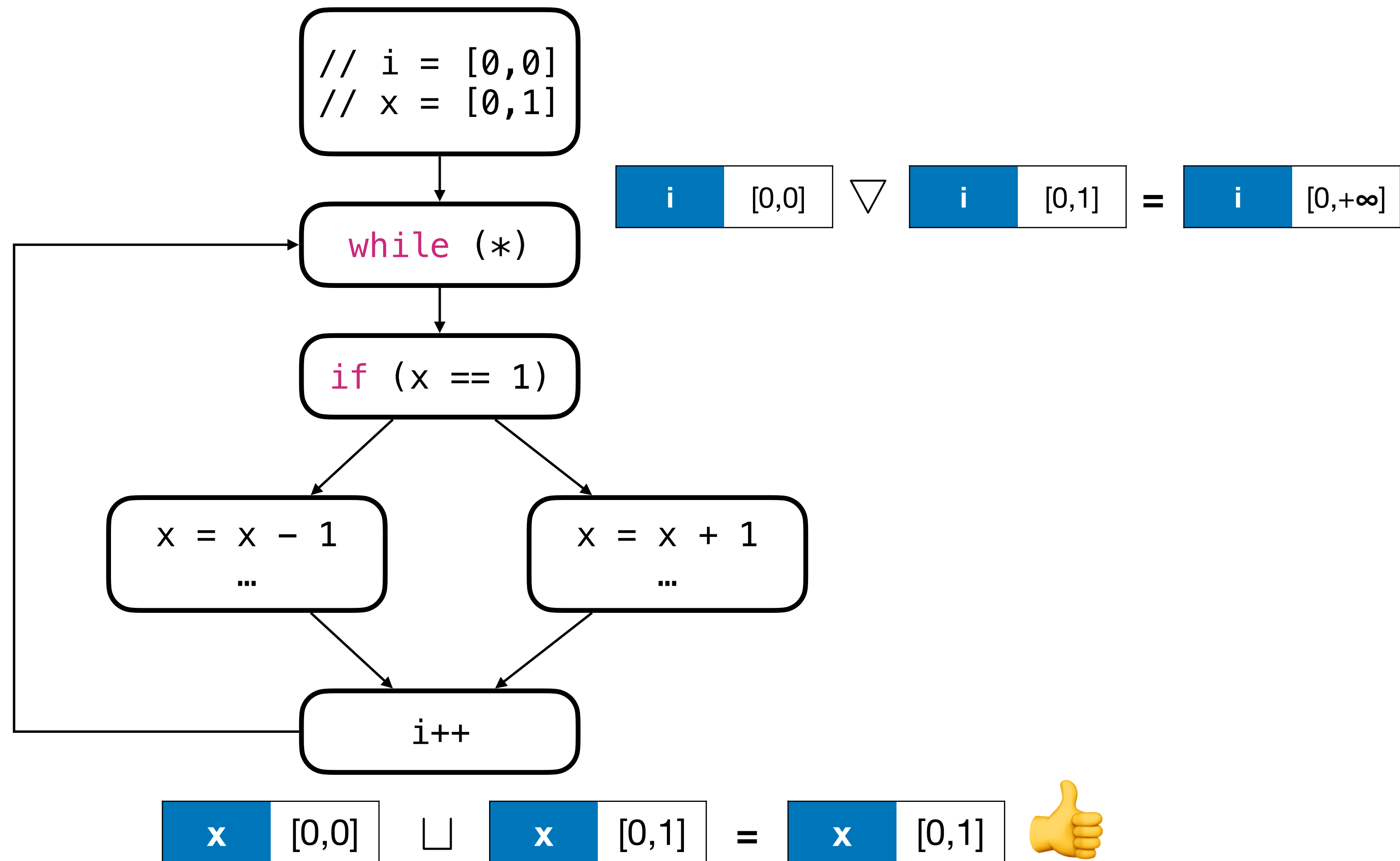
- Consider an analysis with the interval abstract domain



# Solution: Selective Widening

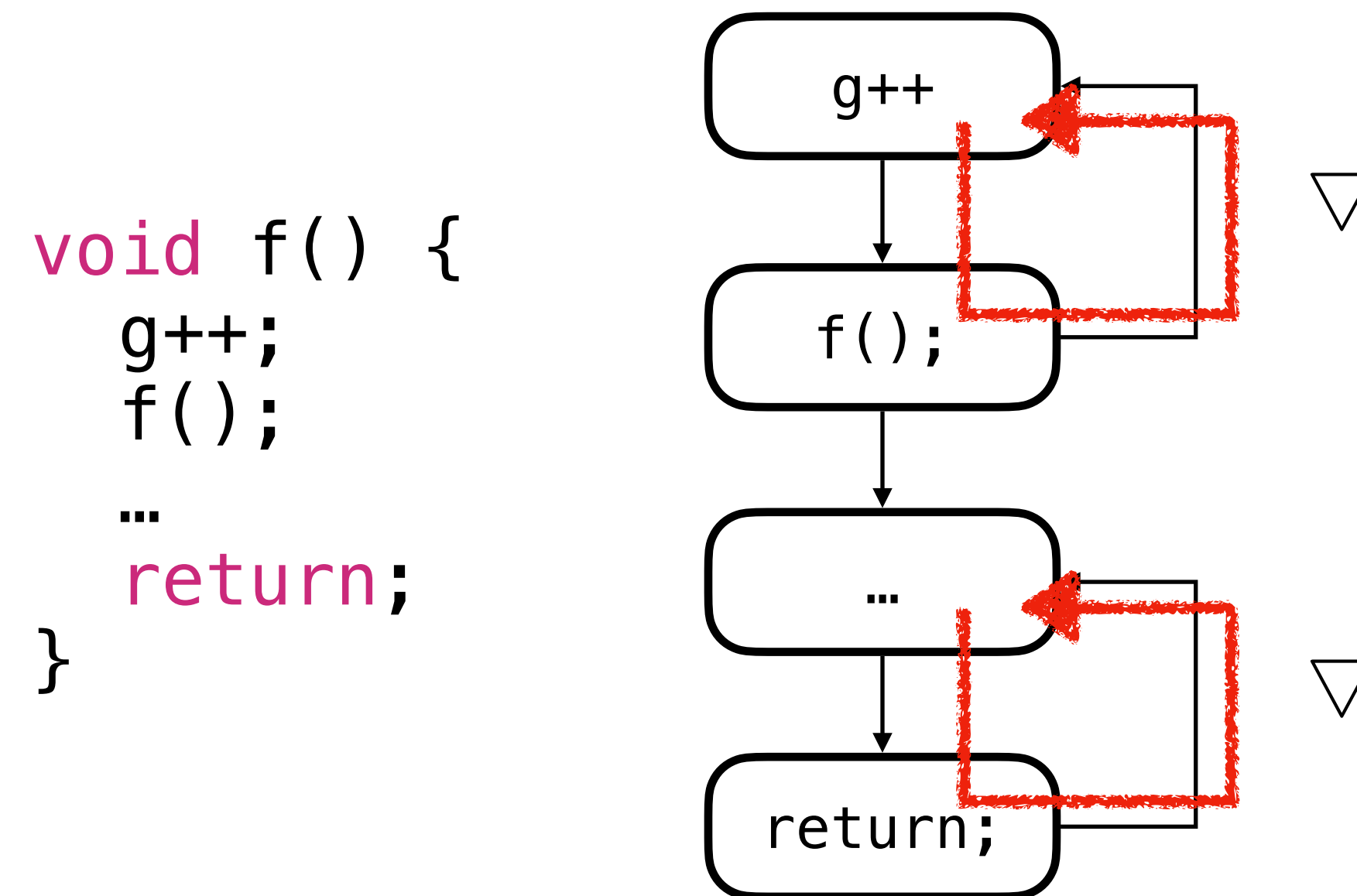
- Apply widening only when the label is the target of a **cycling** control flow
  - e.g., while-loop heads, targets of cycling gotos, (spurious) call-cycle
- For other labels, apply the **join** operation instead

# Case 1: Loop Heads



# Case 2: Call-cycle

- Widening when a **recursive call-cycle** exists

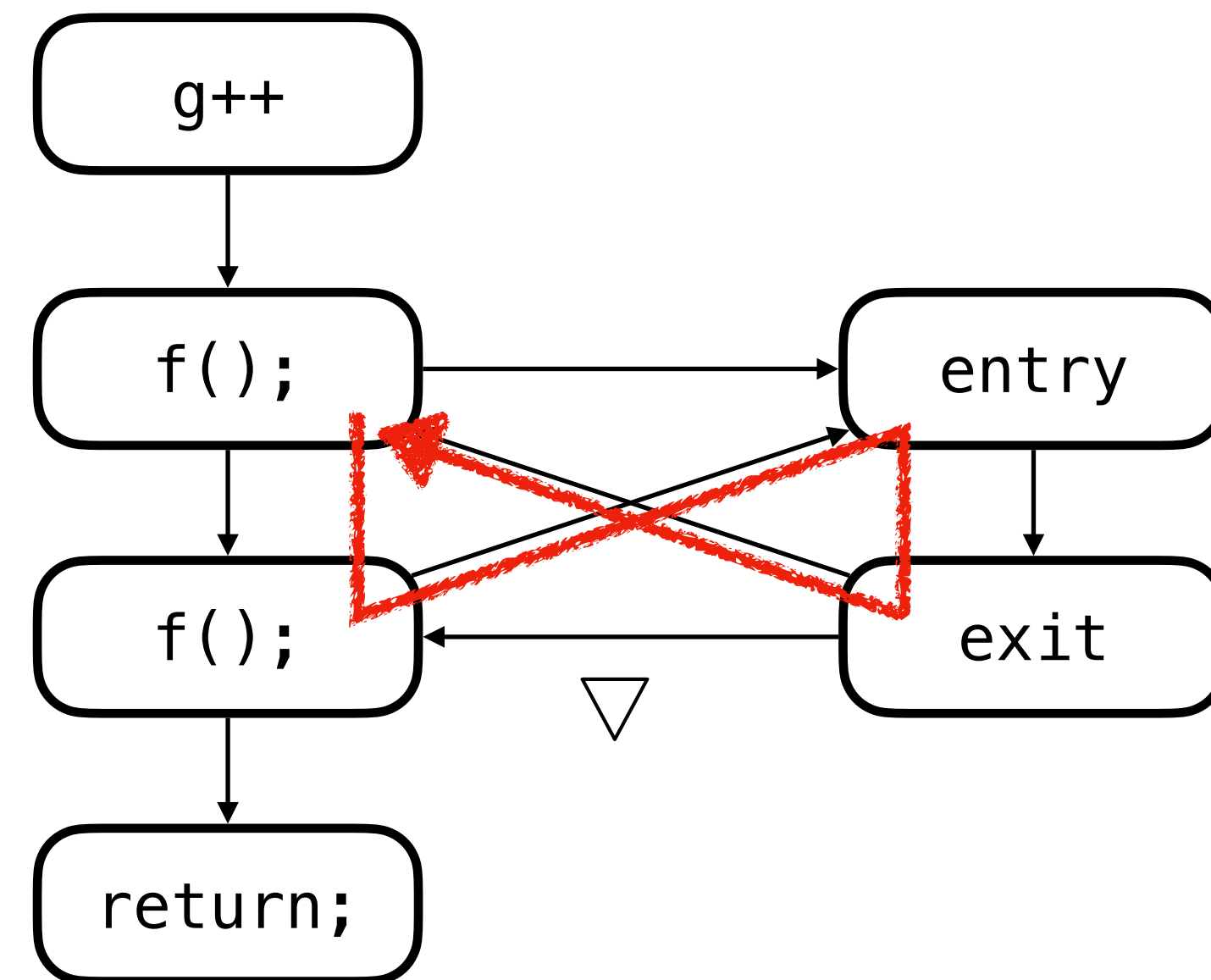




# Case 2: Call-cycle (Cont'd)

- Widening when even **spurious-cycle** happens
  - For example, context-insensitive analysis

```
int main() {  
    g++;  
    f();    // non-recursive  
    f();  
    return;  
}
```



# Caveat

- In general, cycle detection cannot be done before analysis
  - control-flow is **dynamic** (e.g., higher-order functions, exceptions, etc)
- Possible solutions:
  - online cycle-detection (during analysis): precise but costly
  - offline cycle-detection with pre-analysis (before analysis): imprecise but lightweight

# Problem 2: Hasty Join

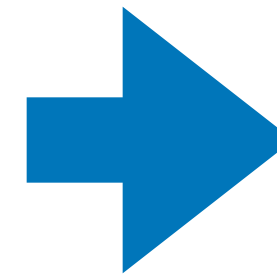
```
x = ?; // any value
i = 1;
while (i > 0) {
  if(x < 0 || x > 1000) {
    x = 0;
  } else {
    x = x + 1;
  }
  input(i);
}
// actually, x is in [0, 1001]
```

Initialization step

- The abstract value for  $x$  with a naive approach would be  $[-\infty, +\infty]$
- Idea: **detach** the first iteration from the rest

# Solution: Loop Unrolling

```
x = ?; // any value
i = 1;
while (i > 0) {
    if(x < 0 || x > 1000) {
        x = 0;
    } else {
        x = x + 1;
    }
    input(i);
}
// actually, x is in [0, 1001]
```



```
x = ?; // any value
i = 1;
if(x < 0 || x > 1000) {
    x = 0;
} else {
    x = x + 1;
}
input(i);
// x is in [0, 1001]
while (i > 0) {
    if(x < 0 || x > 1000) {
        x = 0;
    } else {
        x = x + 1;
    }
    input(i);
}
// x is in [0, 1001]
```

} first iter.

} rest

# Problem 3: Hasty Widening

```
x = 0;
while (*) {
  if(*) {
    x = -1;
  } else {
    x = x + 1;
  }
}
// x >= -1
```

$$\boxed{x \quad [0,0]} \nabla \boxed{x \quad [-1,1]} = \boxed{x \quad [-\infty, +\infty]}$$

- The abstract value of x with a naive approach would be  $[-\infty, +\infty]$
- Idea: **delay** the application of widening for the first N iterations

# Solution: Delayed Widening

```
x = 0;
while (*) {
  if(*) {
    x = -1;
  } else {
    x = x + 1;
  }
}
// x >= -1
```

Delayed widening where  $N = 1$

$$\boxed{x \quad [0,0]} \sqcup \boxed{x \quad [-1,1]} = \boxed{x \quad [-1,1]}$$

$$\boxed{x \quad [-1,1]} \nabla \boxed{x \quad [-1,2]} = \boxed{x \quad [-1,+\infty]}$$

$$\boxed{x \quad [-1,+\infty]} \nabla \boxed{x \quad [-1,+\infty]} = \boxed{x \quad [-1,+\infty]}$$

**Fixed Point!**

# Problem 4: Excessive Widening

```
x = 0;
while (x <= 100) {
  if(x >= 50) {
    x = 10;
  } else {
    x = x + 1;
  }
}
// actually, x is in [0, 50]
```

$$\boxed{x \mid [0,0]} \nabla \boxed{x \mid [0,1]} = \boxed{x \mid [0,+\infty]}$$

- The abstract value of  $x$  with a naive approach is  $[0, +\infty]$
- Idea: use a **slower and more precise** widening

# Solution: Widening with Thresholds

- Take several small steps and stops at pre-defined threshold values
- For example, consider only one threshold  $B$ :

**A naive widening operator**

$$[n, p] \nabla [n, q] = \begin{cases} [n, p] & \text{if } p \geq q \\ [n, +\infty] & \text{if } p < q \end{cases}$$

**A widening with thresholds**

$$[n, p] \nabla [n, q] = \begin{cases} [n, p] & \text{if } p \geq q \\ [n, B] & \text{if } p < q \leq B \\ [n, +\infty] & \text{if } B < q \end{cases}$$

\*only the right bounds, for brevity



# Widening with Thresholds

```
x = 0;  
while (x <= 100) {  
    if(x >= 50) {  
        x = 10;  
    } else {  
        x = x + 1;  
    }  
}
```

Thresholds = {50}

$$\boxed{x \quad [0,0]} \nabla \boxed{x \quad [0,2]} = \boxed{x \quad [0,50]}$$

$$\boxed{x \quad [0,50]} \nabla \boxed{x \quad [0,50]} = \boxed{x \quad [0,50]}$$

**Fixed Point!**

# Summary

- “***Loop is evil***”: one of the main source of imprecision
- Important to design effective iteration techniques
  - no universal solutions
  - depending on the target program’s characteristics
- Need for domain knowledge (human experts or learning techniques)