Introduction to Program Analysis
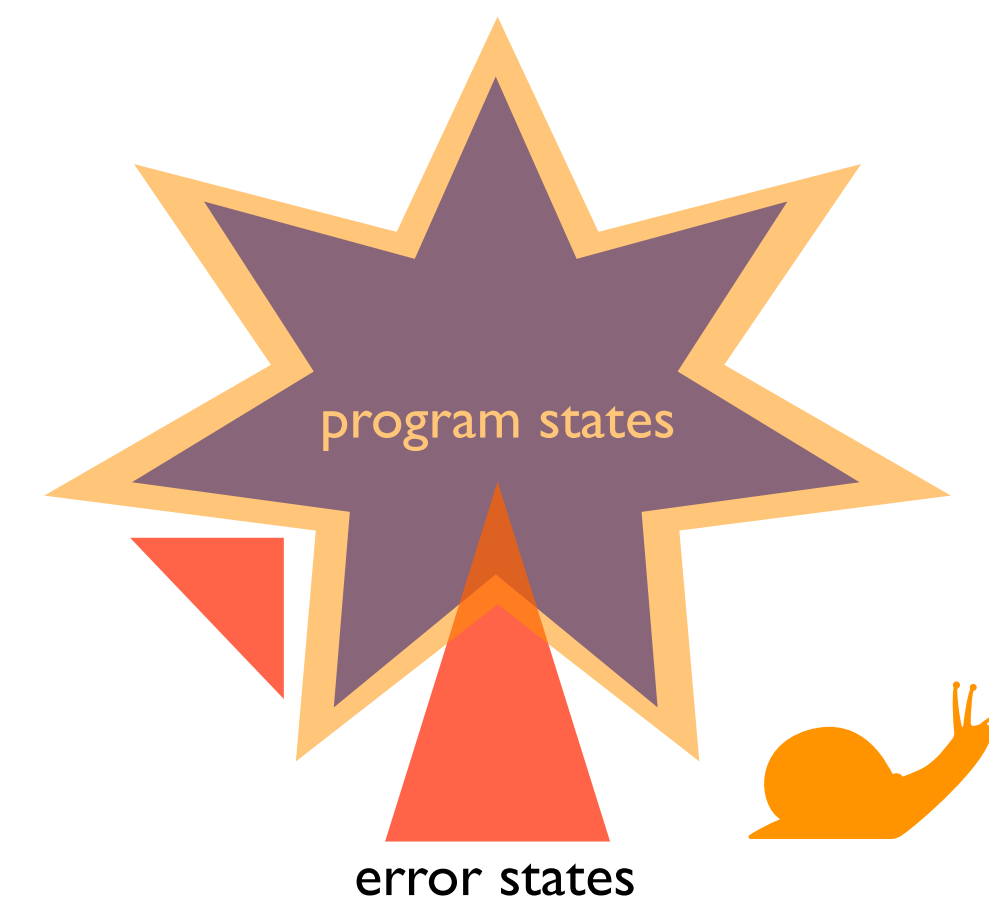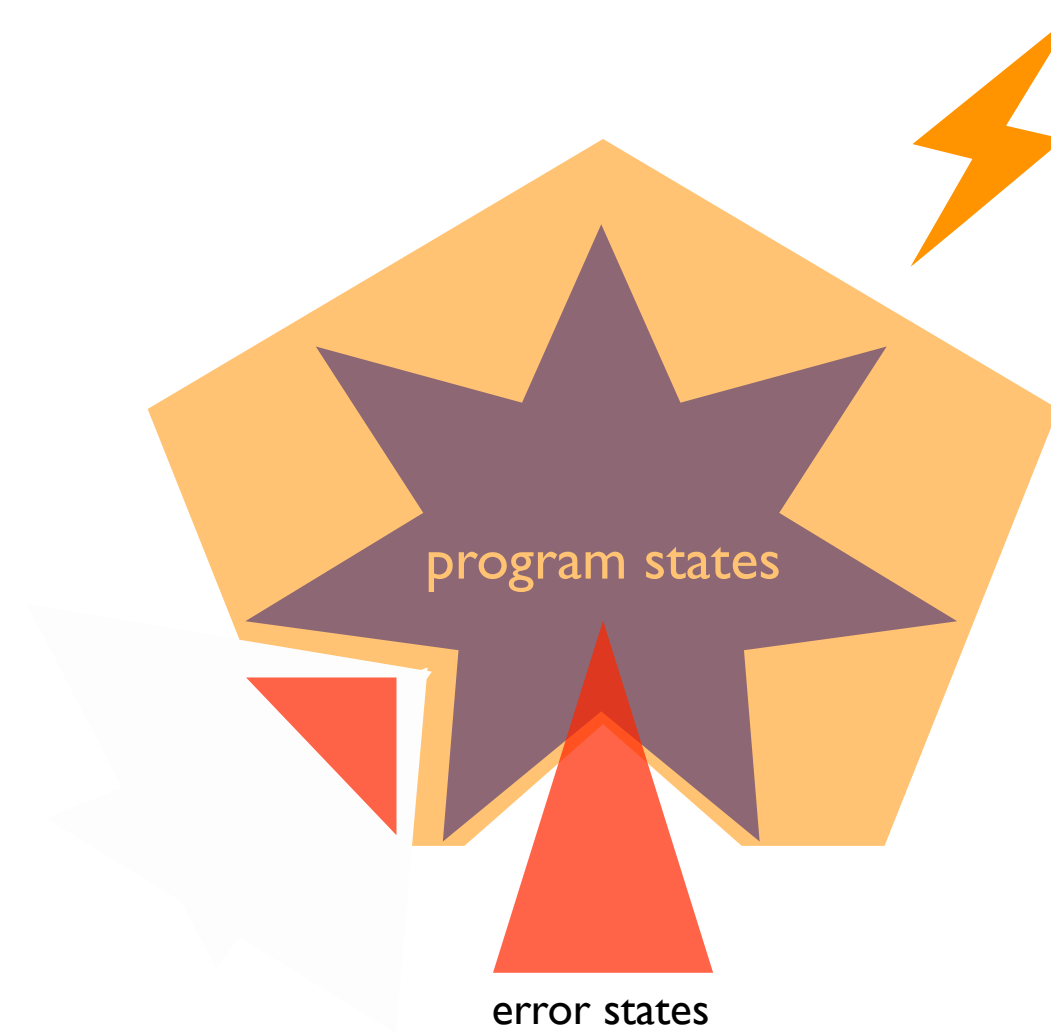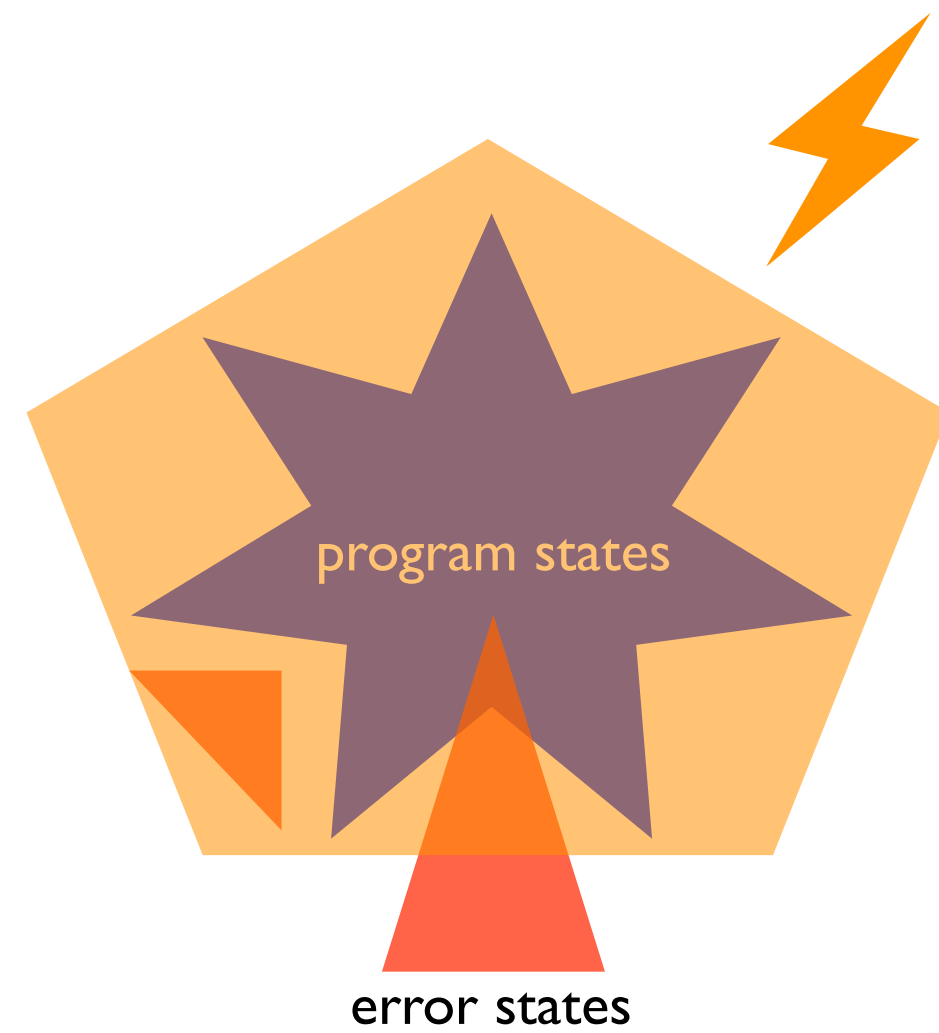
12. Selective X-sensitivity
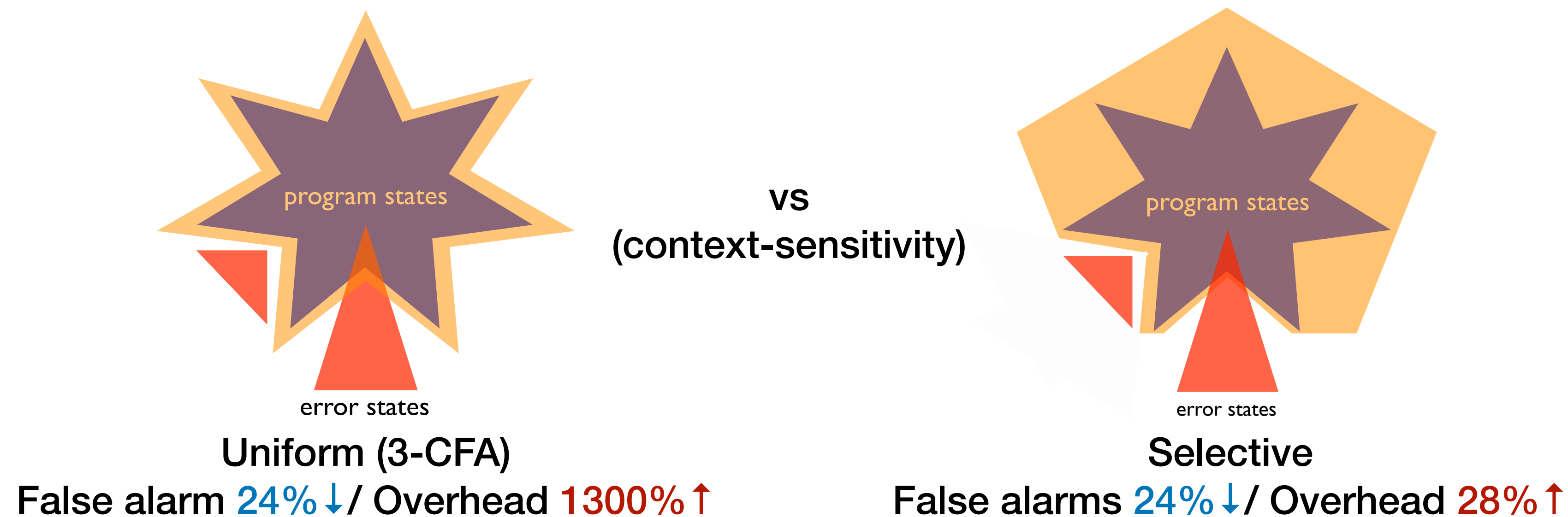
Kihong Heo

KAIST

Cost-Accuracy Balance

- How to **strike a balance** between cost and accuracy?

- How to find a **minimal abstraction** to produce the **best result**?

Selective X-Sensitivity

- Selectively apply a precision-improving technique X-sensitivity

  - only when/where it matters

  - X : context, flow, variable relationship, etc

vs
(context-sensitivity)

Uniform (3-CFA)
False alarm 24%↓ / Overhead 1300%↑

Selective
False alarms 24%↓ / Overhead 28%↑

*Oh et al. Selective Context-Sensitivity Guided by Impact Pre-analysis, PLDI'14

Example: Context-Sensitivity

- Suppose an analysis with the interval domain

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
      assert(p > 1); // Q1: always true
2:    q = h(input());
      assert(q > 1); // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```

Example: Context-Sensitivity

- Context-insensitive analysis


```
int h(n) { return n; }           [-oo, +oo]

void f(s) {
1:  p = h(s);
    assert(p > 1);  // Q1: always true
2:  q = h(input());
    assert(q > 1); // Q2: not always true
}

3: void g() { f(8); }

   void main(){
4:   f(4);
5:   g();
6:   g();
   }
```
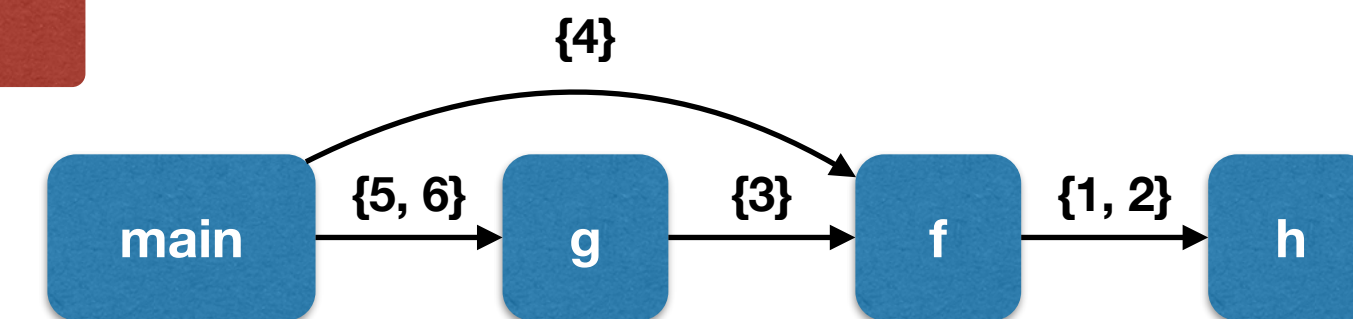
cannot prove

Example: Context-Sensitivity

- Uniformly context-sensitive analysis (3-CFA)

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
      assert(p > 1);  // Q1: always true
2:    q = h(input());
      assert(q > 1);  // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```


Value of n

Example: Context-Sensitivity
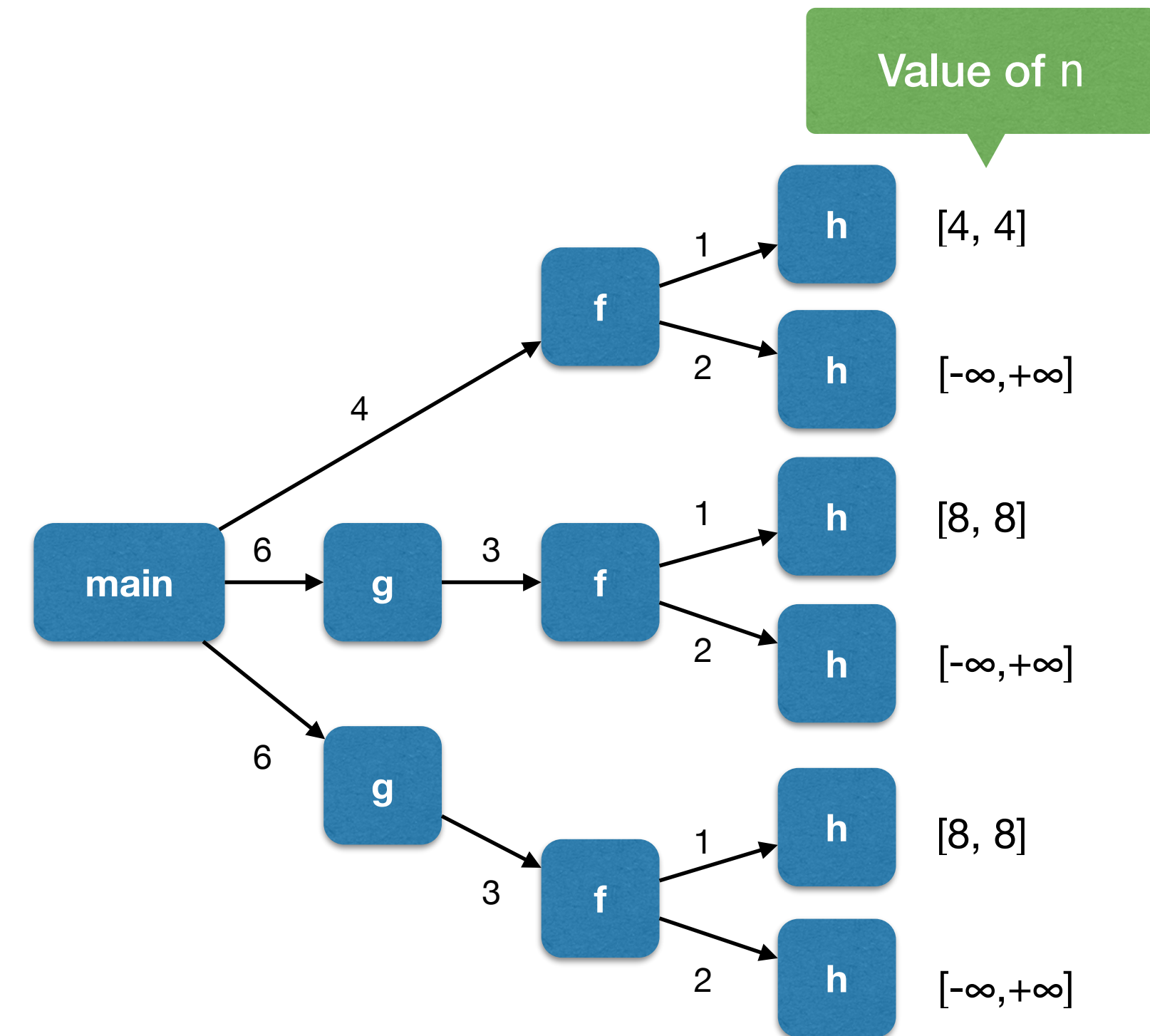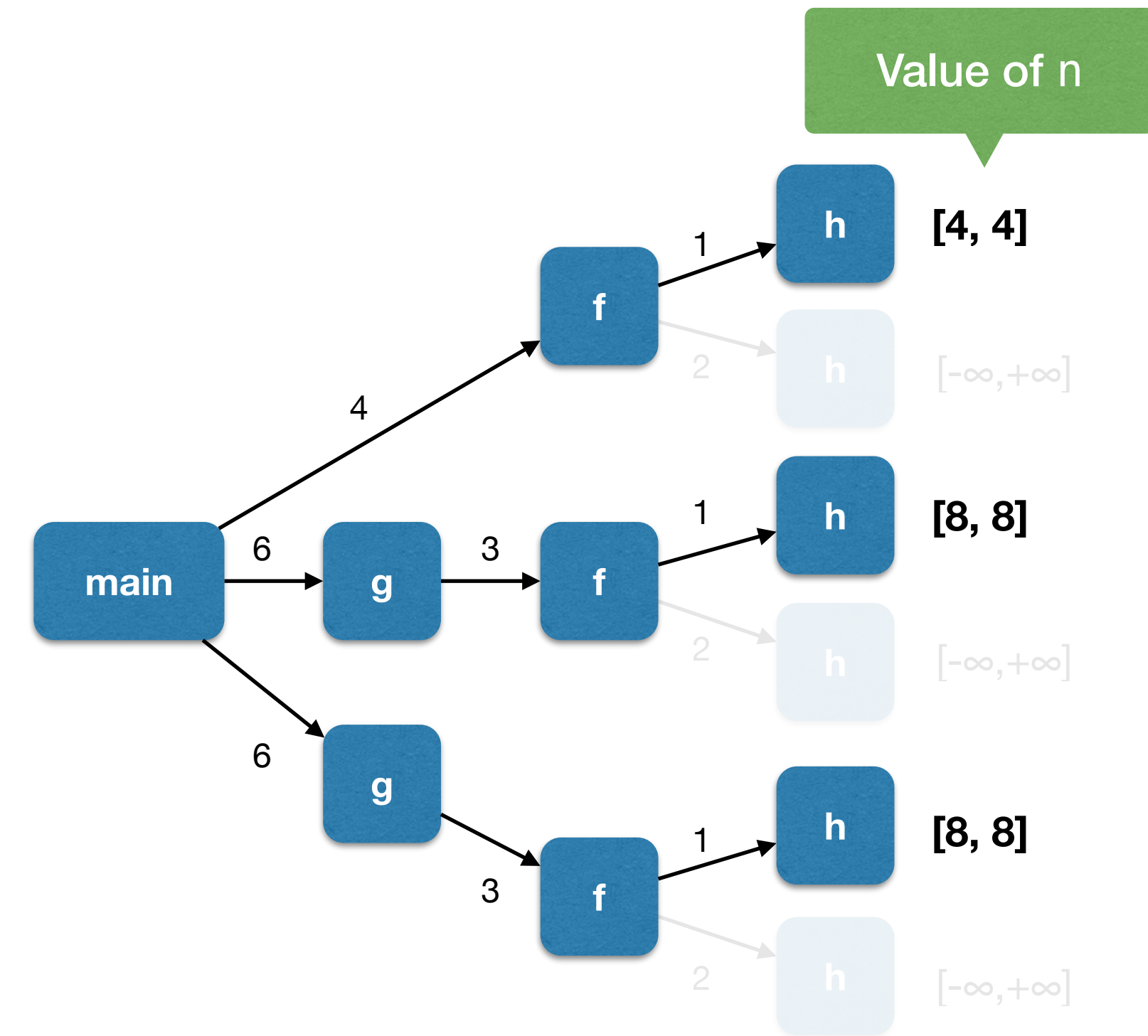
- Uniformly context-sensitive analysis (3-CFA)

```
      int h(n) { return n; }

      void f(s) {
1:      p = h(s);
        assert(p > 1);  // Q1: always true
2:      q = h(input());
        assert(q > 1);  // Q2: not always true
      }

3: void g() { f(8); }

      void main(){
4:      f(4);
5:      g();
6:      g();
      }
```

Proved!

Value of n

Example: Context-Sensitivity
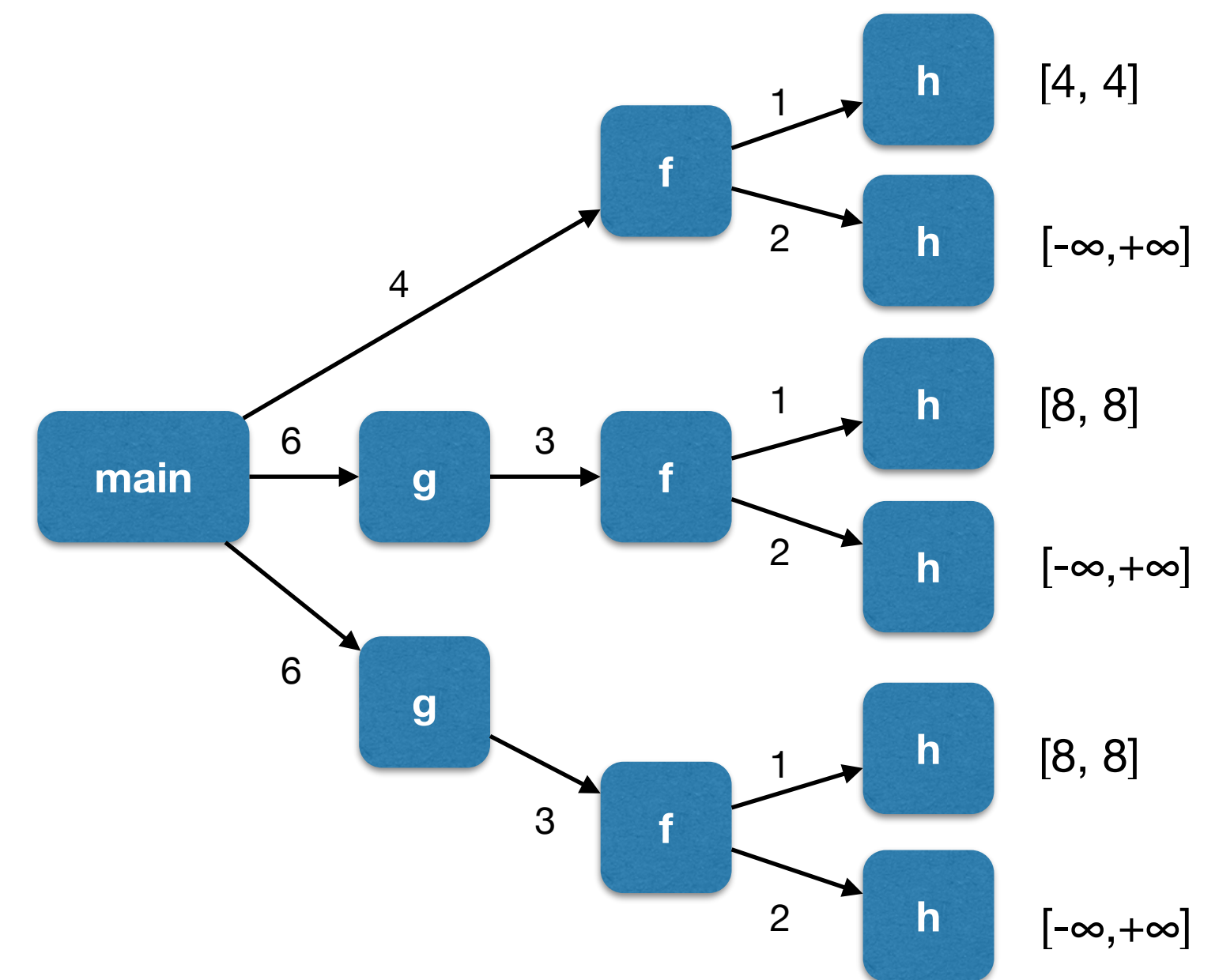
- Problem of uniformly sensitive analysis

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
      assert(p > 1); // Q1: always true
2:    q = h(input());
      assert(q > 1); // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```
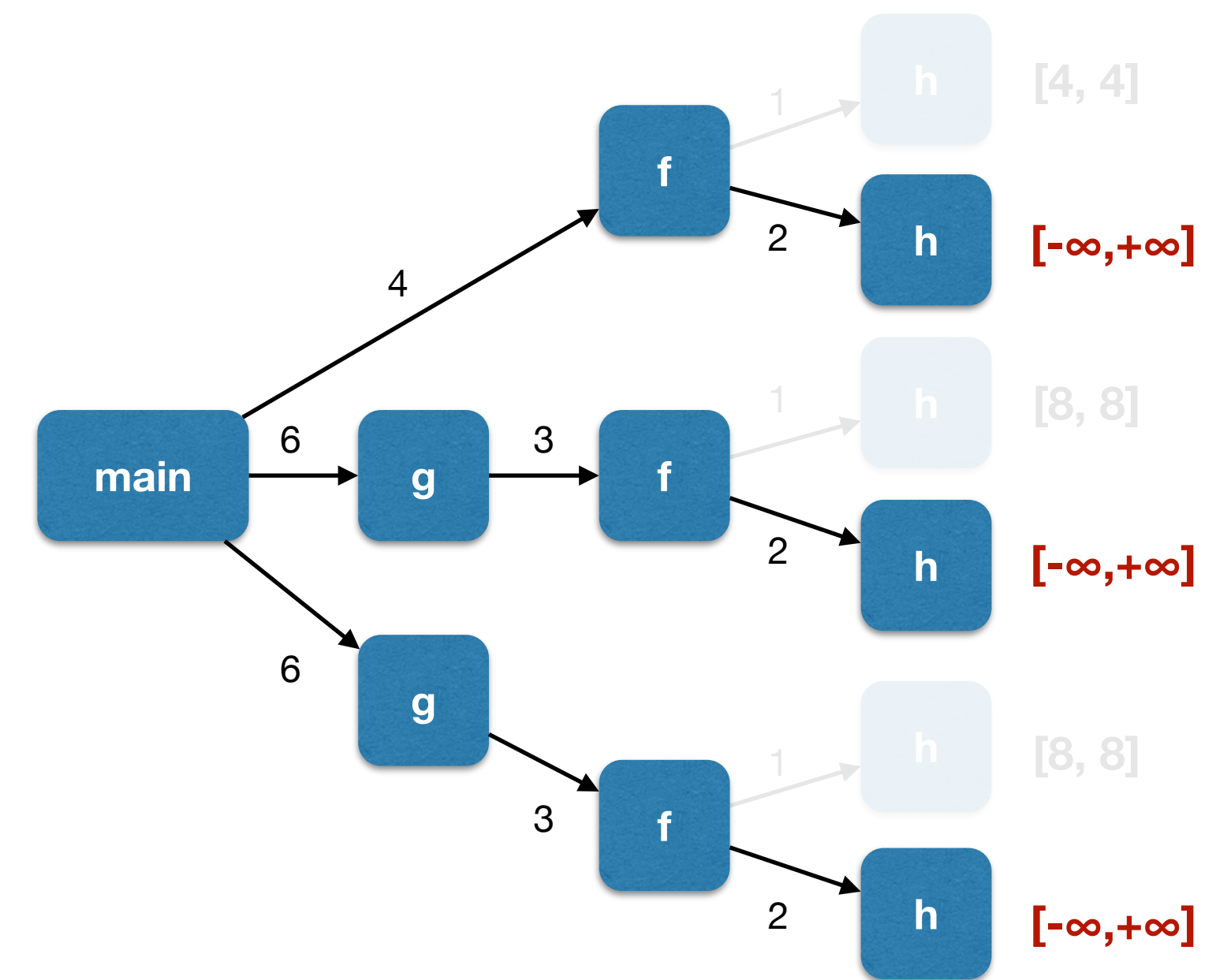
Example: Context-Sensitivity

- Problem of uniformly sensitive analysis 1: useless sensitivity

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
      assert(p > 1);  // Q1: always true
2:    q = h(input());
      assert(q > 1);  // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```
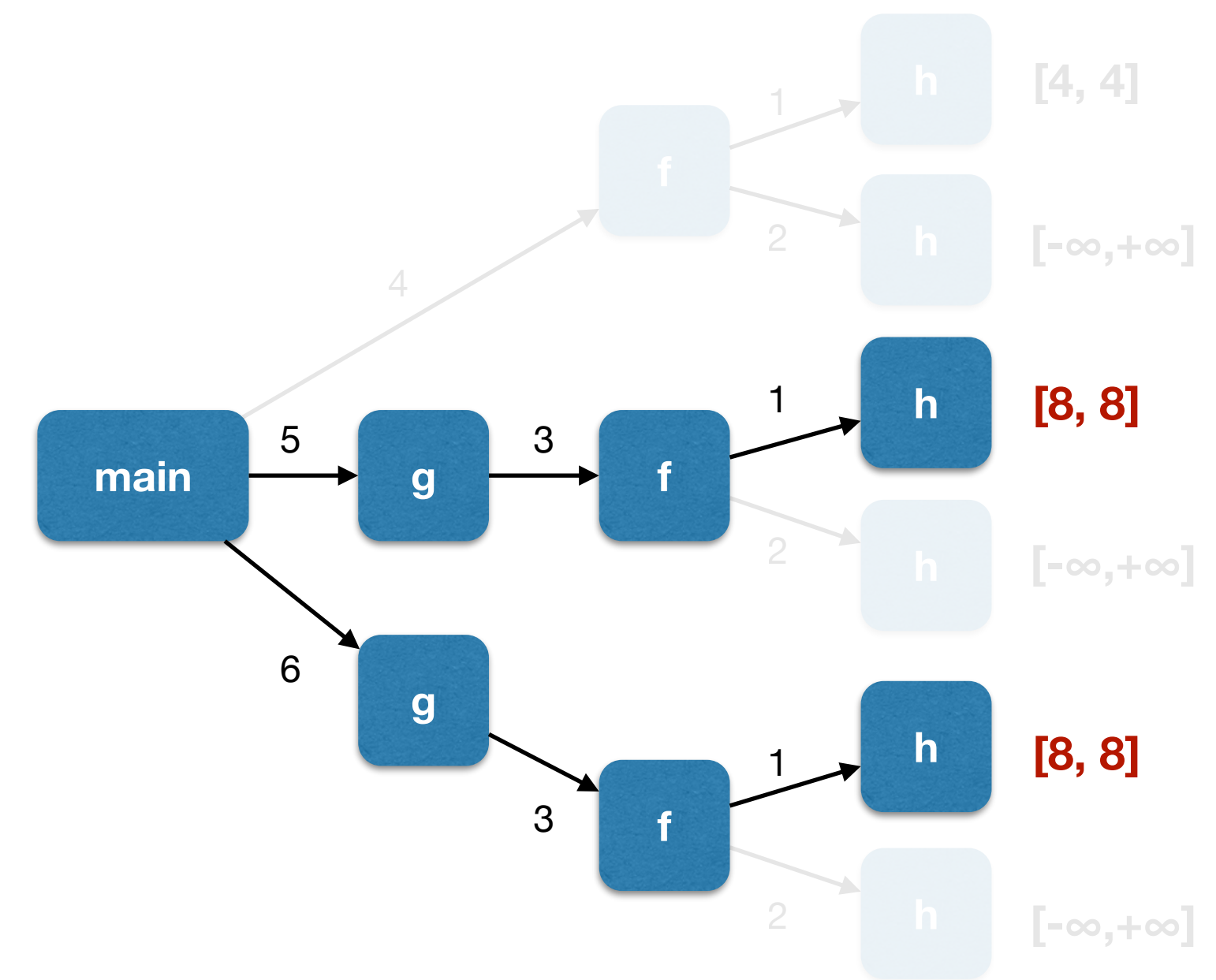
Example: Context-Sensitivity

- Problem of uniformly sensitive analysis 2: too much sensitivity

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
      assert(p > 1); // Q1: always true
2:    q = h(input());
      assert(q > 1); // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```

Example: Context-Sensitivity

- Solution: **selective** context-sensitive analysis

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
      assert(p > 1); // Q1: always true
2:    q = h(input());
      assert(q > 1); // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```
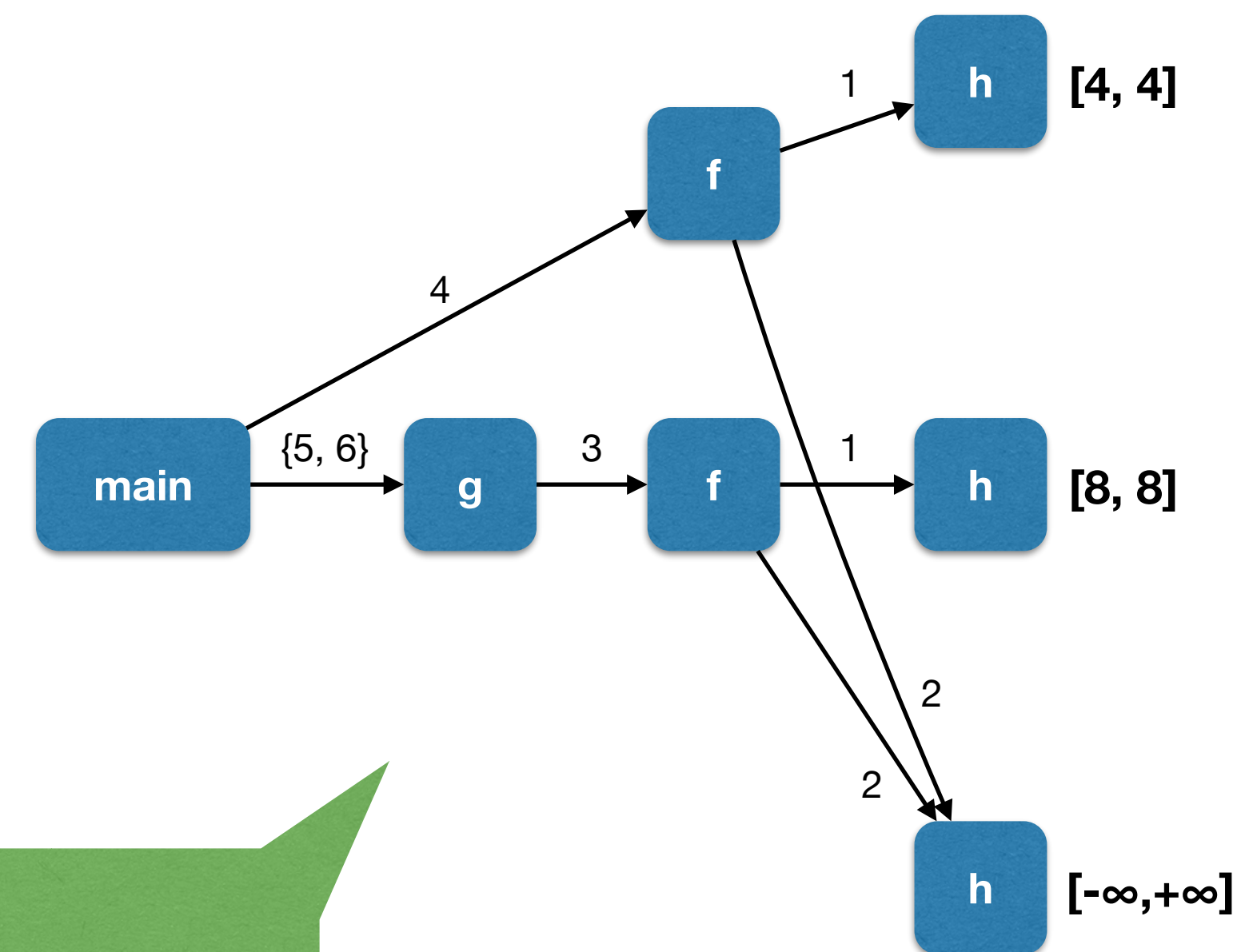
Q: How to infer this selective context-sensitivity?

Key Idea: Impact Pre-Analysis

- Estimate the impact of X-sensitivity on main analysis

  - Fully X-sensitive, but approximated in other precision aspects

Design of Impact Pre-Analysis

- Main analysis: context-insensitive + interval domain

- Impact pre-analysis: fully context-sensitive + approximated interval domain

$\top$ ← All intervals

★ ← non-negative intervals
e.g., [5,7], [0, +oo]

$\bot$

$$\wp(\mathbb{Z}^{\sharp}) \xleftrightarrow[\alpha]{\gamma} \{\bot, \bigstar, \top\}$$

Running Impact Pre-Analysis

- Impact pre-analysis: fully context-sensitive + approximated interval domain

```
    int h(n) { return n; }

    void f(s) {
1:     p = h(s);
       assert(p > 1); // Q1: always true
2:     q = h(input());
       assert(q > 1); // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:     f(4);
5:     g();
6:     g();
    }
```
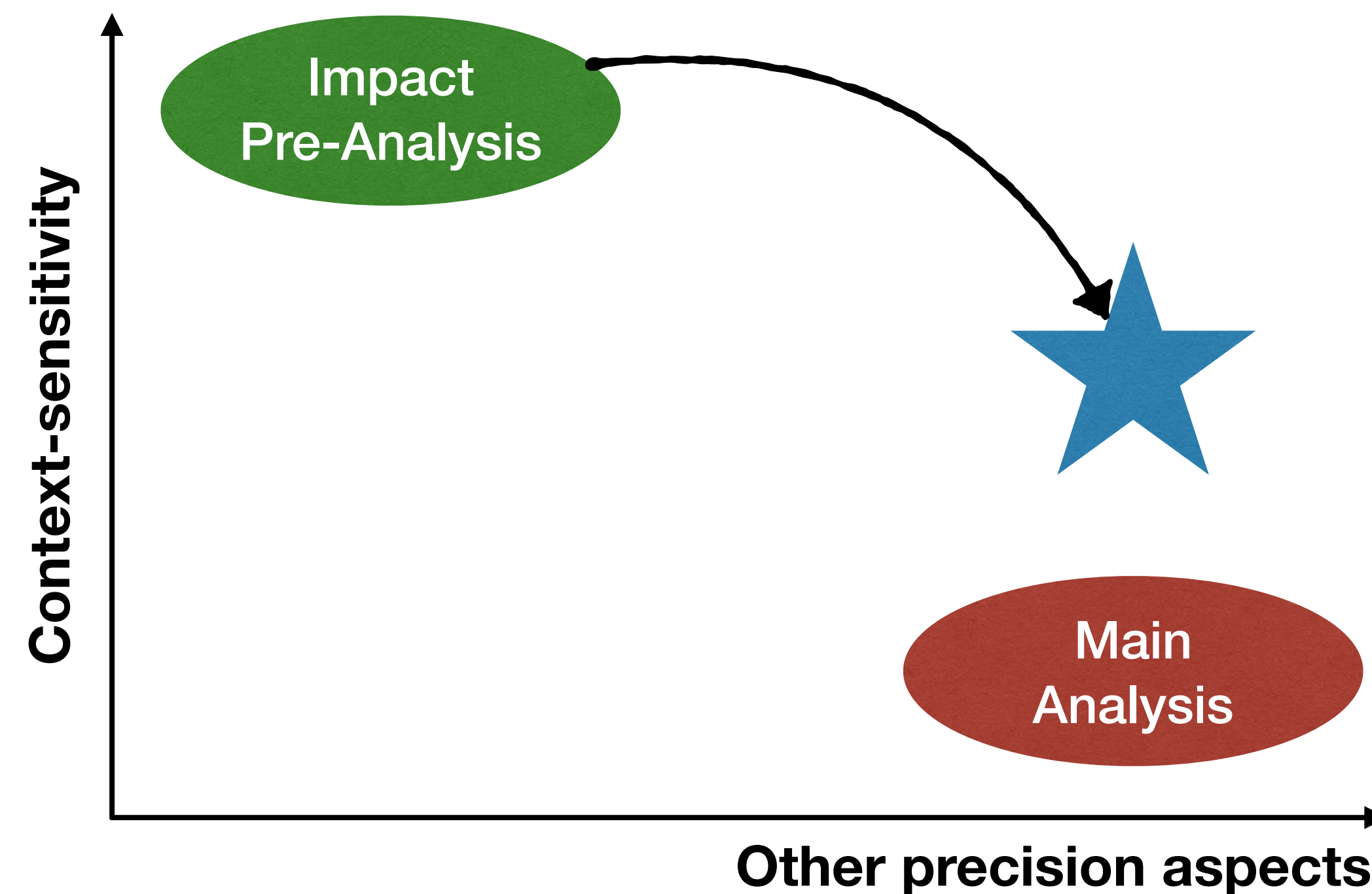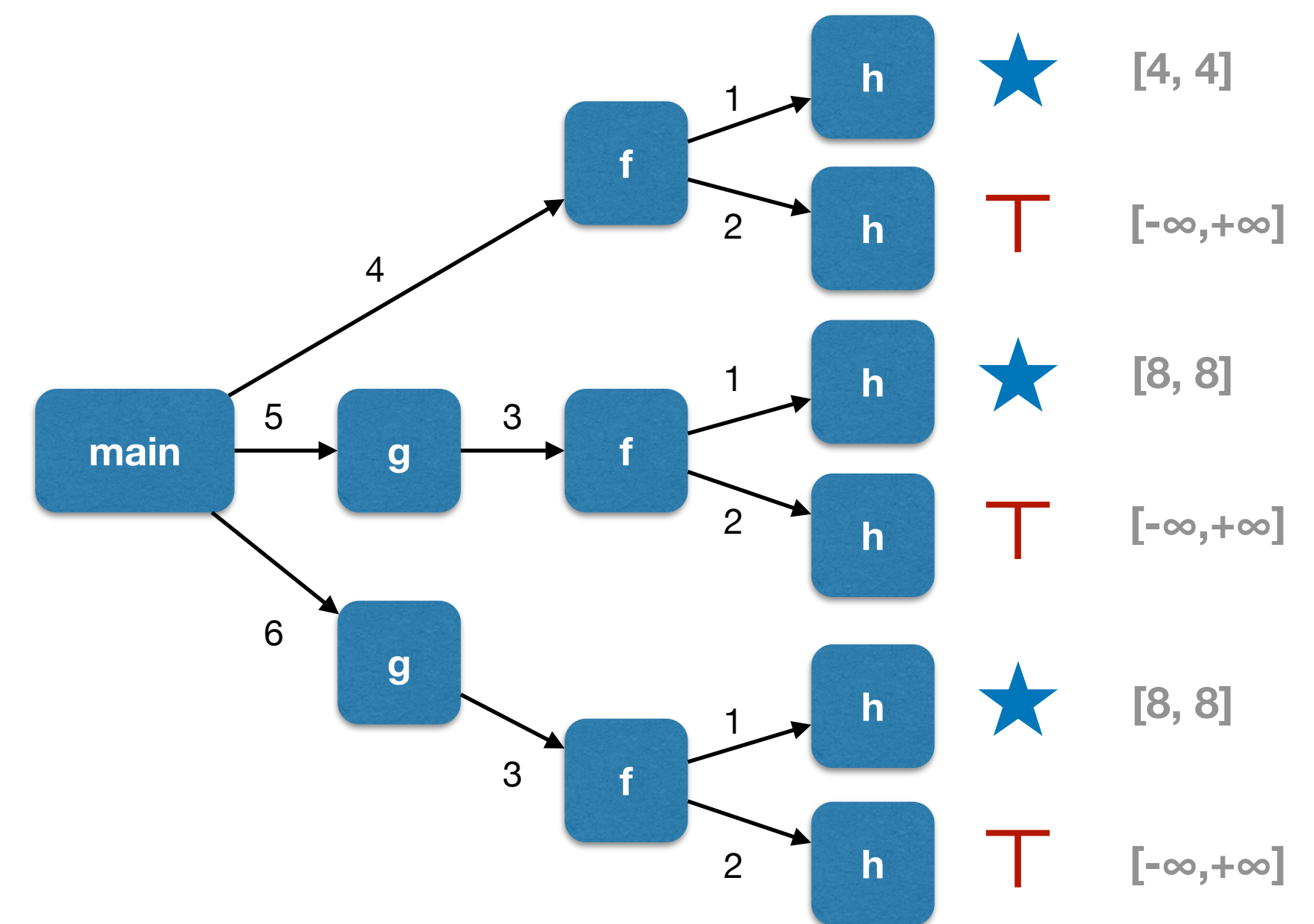
Constructing Selective Sensitivity

1. Collect queries whose expressions are assigned with ★

```
    int h(n) { return n; }

    void f(s) {
1:    p = h(s);
  ★ assert(p > 1);  // Q1: always true
2:    q = h(input());
  ⊤ assert(q > 1);  // Q2: not always true
    }

3: void g() { f(8); }

    void main(){
4:    f(4);
5:    g();
6:    g();
    }
```

Constructing Selective Sensitivity
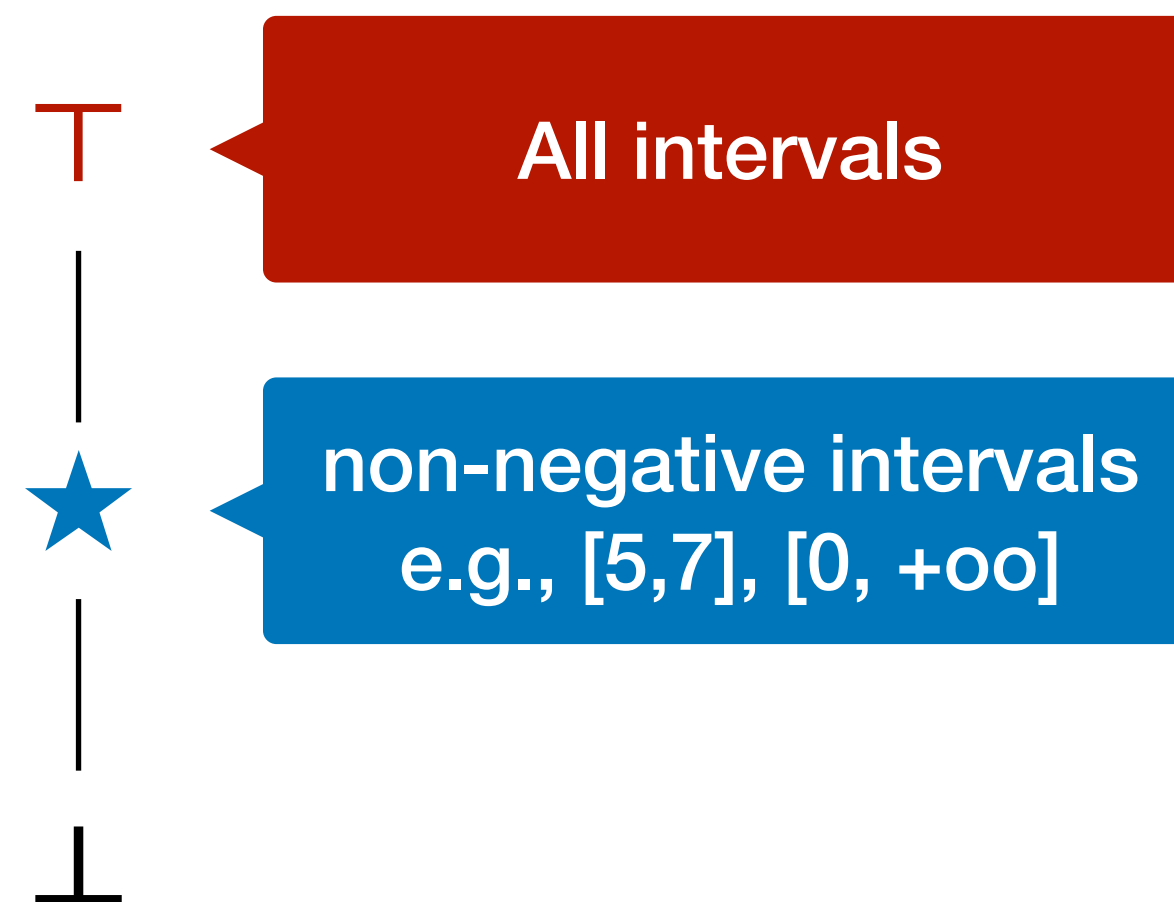
2. Find contexts that contribute to the selected queries

```
     int h(n) { return n; }

     void f(s) {
1:     p = h(s);
   ★   assert(p > 1); // Q1: always true
2:     q = h(input());
   ⊤   assert(q > 1); // Q2: not always true
     }

3: void g() { f(8); }

     void main(){
4:     f(4);
5:     g();
6:     g();
     }
```
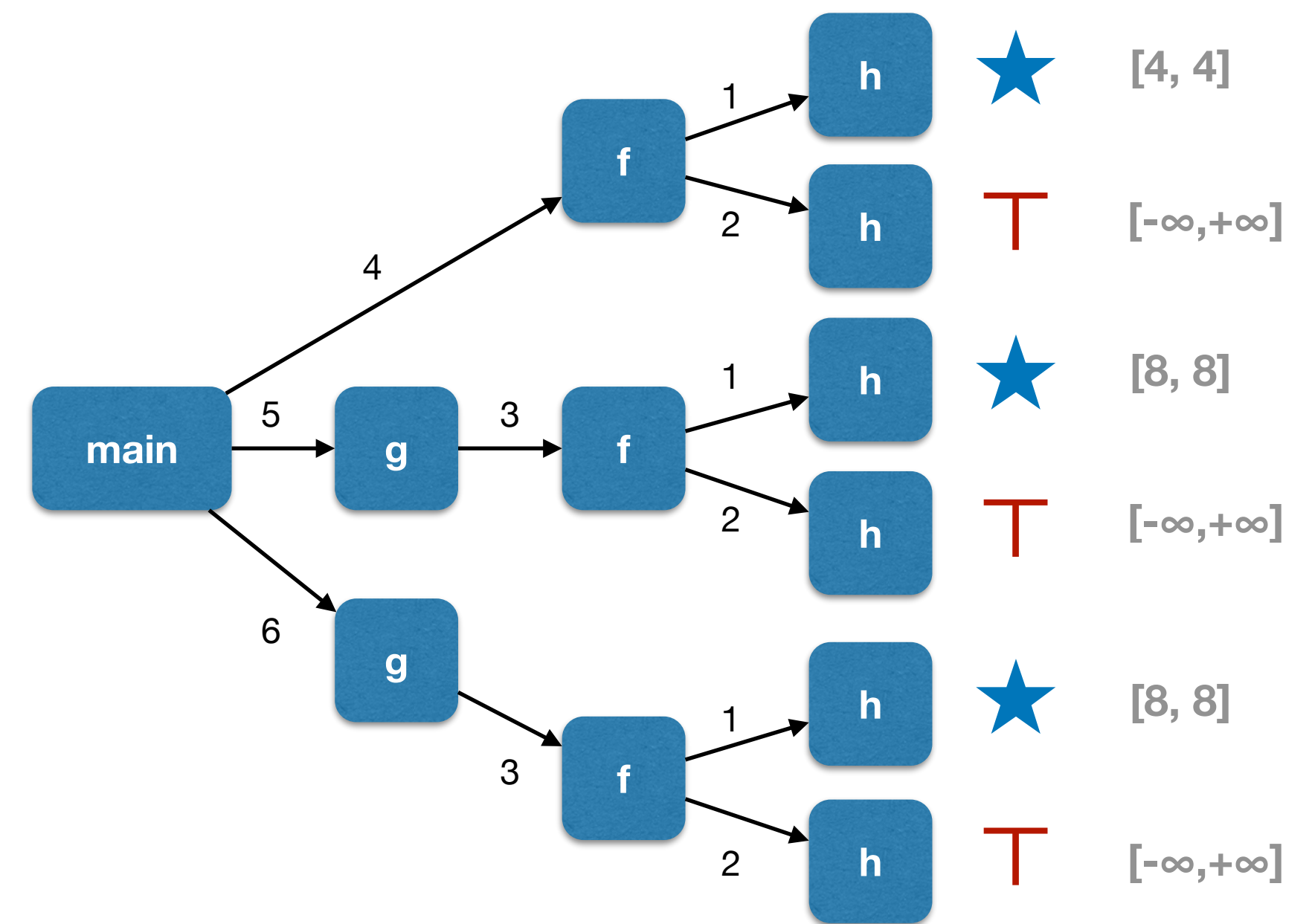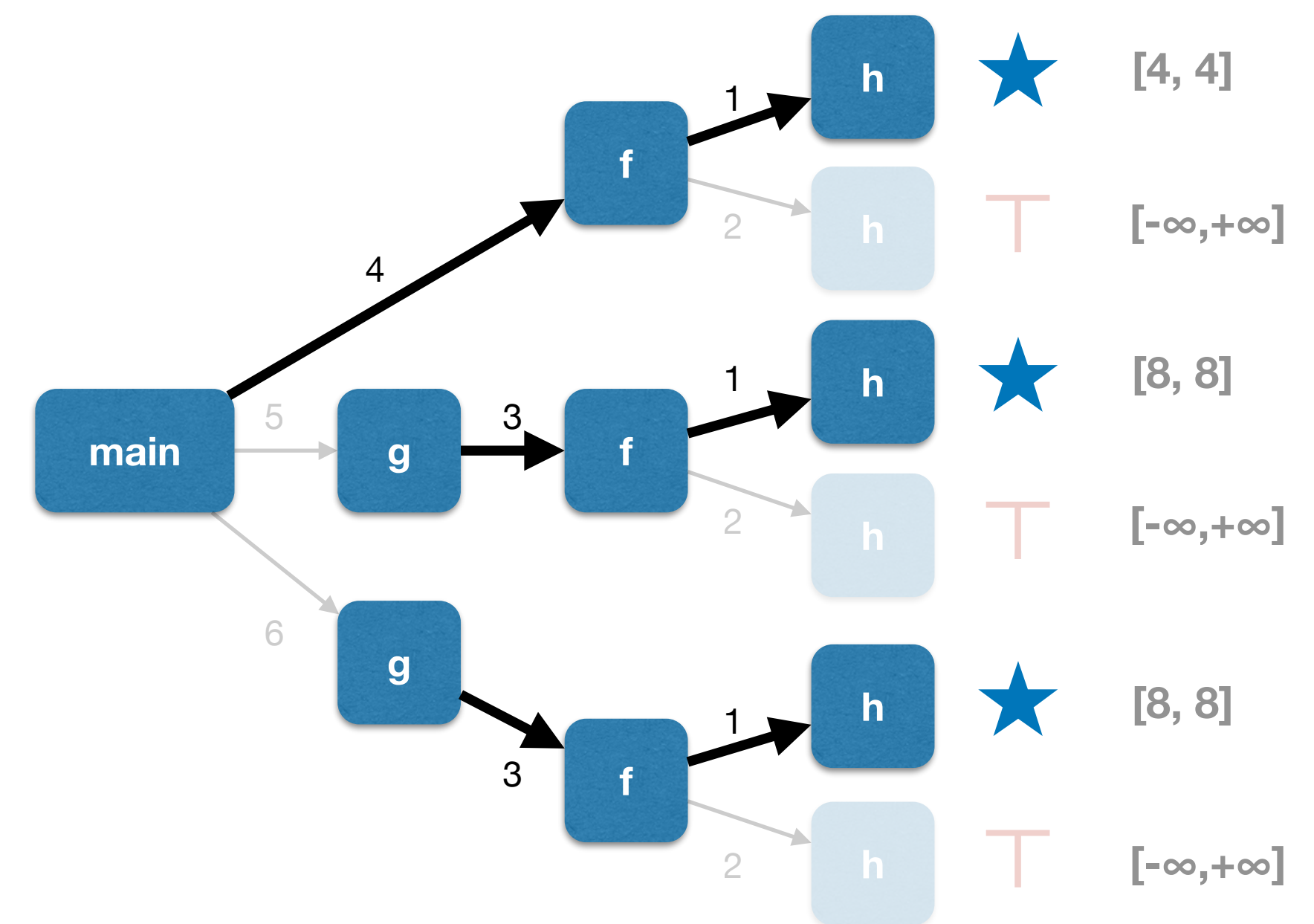

Selected contexts for function h: {4·1, 3·1}

Generality

- The same principle is applicable to other types of sensitivity

  - Running an impact pre-analysis
    (full X-sensitivity + aggressive abstraction of other aspects)

  - Select queries that are judged promising by the pre-analysis

  - Construct X-sensitivity that contributes to the selected queries

Example: Relational Analysis

- Keep track of relationships between variables in a certain form

  - E.g., octagon analysis: $(\pm x) - (\pm y) \leq c$

Intervals   Octagons   Polyhedra
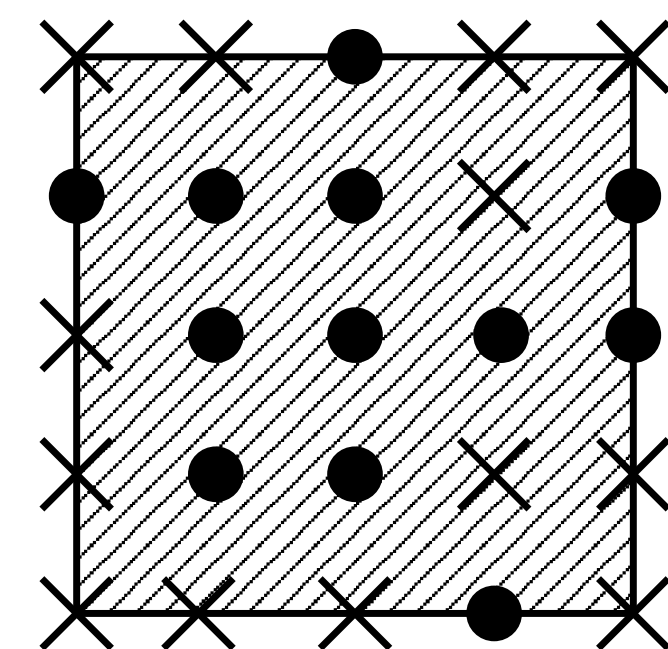
Non-relational
Abstraction
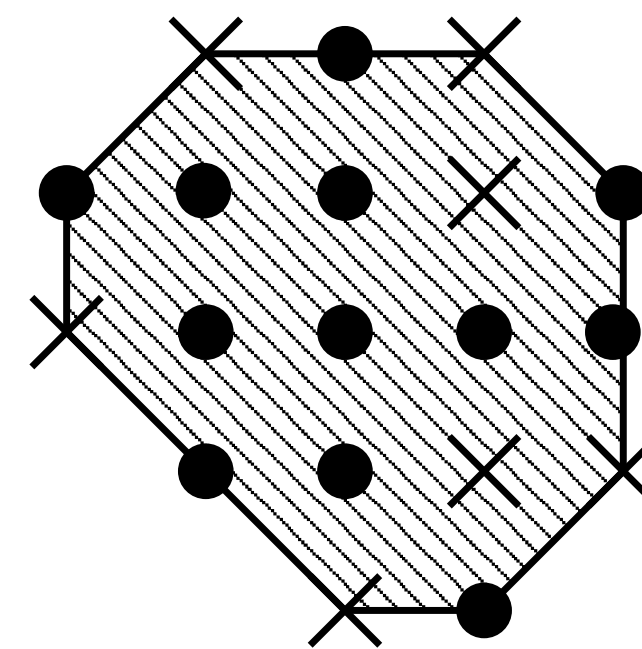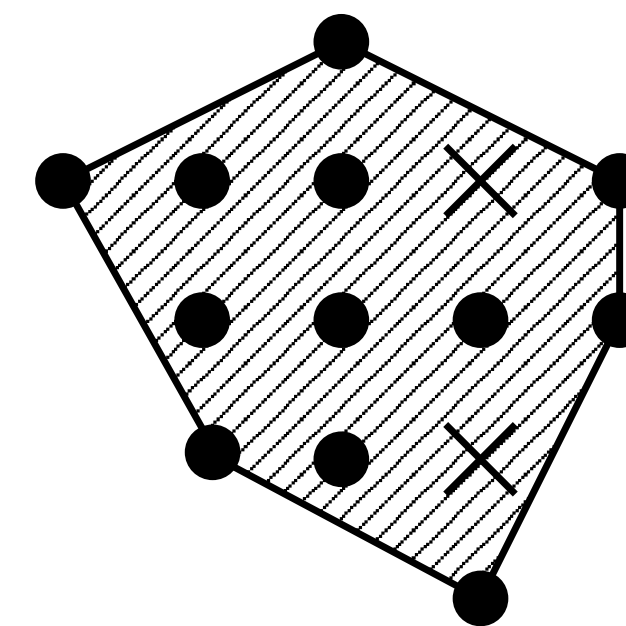
Relational
Abstraction

*A. Mine, The Octagon Abstract Domain, HOSC'06

Example: Relational Analysis

- Non-relational analysis with the interval domain

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();          // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a);          // Q1: always true
5:   assert(i < c);          // Q2: not always true
6: }
```

Var	Val
a	[-oo, +oo]
b	[-oo, +oo]
c	[-oo, +oo]
i	[0, +oo]

Example: Relational Analysis

- Fully relational analysis with the octagon domain: $(\pm x) - (\pm y) \leq c$

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();          // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a);          // Q1: always true
5:   assert(i < c);          // Q2: not always true
6: }
```

	a	b	c	i
a	0	∞	∞	∞
b	∞	0	∞	∞
c	∞	∞	0	∞
i	∞	∞	∞	0

{a, b, c, i}

*Consider x - y ≤ c only, for simplicity

Example: Relational Analysis

- Fully relational analysis with the octagon domain: $(\pm x) - (\pm y) \leq c$

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();          // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a);          // Q1: always true
5:   assert(i < c);          // Q2: not always true
6: }
```


	a	b	c	i
a	0	0	∞	∞
b	0	0	∞	∞
c	∞	∞	0	∞
i	∞	∞	∞	0

a - b ≤ 0
b - a ≤ 0

{a, b, c, i}

*Consider x - y ≤ c only, for simplicity

Example: Relational Analysis

- Fully relational analysis with the octagon domain: $(\pm x) - (\pm y) \leq c$

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();          // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a);          // Q1: always true
5:   assert(i < c);          // Q2: not always true
6: }
```

	a	b	c	i
a	0	0	∞	∞
b	0	0	∞	∞
c	∞	∞	0	∞
i	∞	∞	∞	0

c - a ≤ ∞
c - b ≤ ∞

a - c ≤ ∞
b - c ≤ ∞

{a, b, c, i}

*Consider x - y ≤ c only, for simplicity

Example: Relational Analysis

- Fully relational analysis with the octagon domain: $(\pm x) - (\pm y) \leq c$

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();        // User input
3: for (i = 0; i < b; i++) {
4:    assert(i < a);        // Q1: always true
5:    assert(i < c);        // Q2: not always true
6: }
```

	a	b	c	i
a	0	0	∞	∞
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

i - b ≤ -1

{a, b, c, i}

*Consider x - y ≤ c only, for simplicity

Example: Relational Analysis

- Fully relational analysis with the octagon domain: $(\pm x) - (\pm y) \leq c$

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();           // User input
3: for (i = 0; i < b; i++) {
4:    assert(i < a);          // Q1: always true
5:    assert(i < c);          // Q2: not always true
6: }
```


	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

b - a ≤ 0
∧ i - b ≤ -I
⇒ i - a ≤ -I

{a, b, c, i}

*Consider x - y ≤ c only, for simplicity

Example: Relational Analysis

- Problem of fully relational analysis: **useless** relationship

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();          // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a);          // Q1: always true
5:   assert(i < c);          // Q2: not always true
6: }
```

	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

{a, b, ~~c~~, i}

*Consider x - y ≤ c only, for simplicity

Example: Relational Analysis

- Solution: **selective** relational analysis

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();        // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a);        // Q1: always true
5:   assert(i < c);        // Q2: not always true
6: }
```

	a	b	i
a	0	0	-1
b	0	0	-1
i	∞	∞	0

{a, b, i}

+

Var	Val
c	[-oo, +oo]

{c}

*Consider x - y ≤ c only, for simplicity

Design of Impact Pre-Analysis

- Main analysis: non-relational analysis

- Impact pre-analysis: fully relational analysis + approximated upper bound

Fully relational
octagon analysis

Fully relational
impact pre-analysis

Maybe infinite

Definitely finite

Constructing Selective Sensitivity

1. Collect queries whose expressions are assigned with ★

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();        // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a)  ★       // Q1: always true
5:   assert(i < c); ⊤       // Q2: not always true
6: }
```

	a	b	c	i
a	★	★	⊤	★
b	★	★	⊤	★
c	⊤	⊤	★	⊤
i	⊤	⊤	⊤	★

Constructing Selective Sensitivity

2. Find variable relationships that contribute to the selected queries

```
// b = [-oo, +oo]
1: int a = b;
2: int c = input();        // User input
3: for (i = 0; i < b; i++) {
4:   assert(i < a); ★       // Q1: always true
5:   assert(i < c); ⊤       // Q2: not always true
6: }
```

	a	b	c	i
a	★	★	⊤	★
b	★	★	⊤	★
c	⊤	⊤	★	⊤
i	⊤	⊤	⊤	★

Selected variables: {a, b, i}

Summary

- Selective X-sensitivity: a framework for balancing between **cost & accuracy**

- Key idea: **"Apply X-sensitivity only when it matters"**

- Estimate the impact of X using the **impact pre-analysis**

  - Full X-sensitivity + aggressive approximation of other precision aspects

- Construct selective X-sensitivity from the guidance of the impact estimation