Introduction to Static Program Analysis with Numerical Abstract Interpretation

Kenny Ballou

Boise State University
Boise, Idaho
United States of America

June 27th 2023

Outline

About Me

- Graduate Student at Boise State University
- Previously Software Developer and Site Reliability Engineer
- Father
- Author
- Blogger: `https://kennyballou.com`

About You

About You

Beginner No idea what static analysis is.

About You

Beginner	No idea what static analysis is.
Intermediate	Heard of it, but fuzzy on its usage.

About You

Beginner	No idea what static analysis is.
Intermediate	Heard of it, but fuzzy on its usage.
Advanced	Know it and use it.

About You

Beginner	No idea what static analysis is.
Intermediate	Heard of it, but fuzzy on its usage.
Advanced	Know it and use it.
Expert	Use and develop new analyses.

Example of Static Analysis

Does this program terminate?

```
public int example(int n) {
    int p0 = 0;
    int p1 = 1;
    int p = 0;

    for (int i = 0; i < n; i++) {
        p = p0 + p1;
        p0 = p1;
        p1 = p;
    }

    return p;
}
```

Example of Static Analysis

How about this program?

```
public int example(int x) {
    int n = x * -1;
    while (true) {
        x--;
        if (x <= 0) {
            return n;
        }
    }
}
```

Example of Static Analysis

Finally, how about this program?

```
public int example(Queue queue) {
    while (!queue.empty()) {
        Object work = queue.deque();
        // process work
    }
    return 0;
}
```

Static analysis computes "facts" about programs

Definitions

program Loosely used to mean function, group of functions, class, or entire application

static analysis Computing of facts about programs without executing them

Static analysis can be used to answer questions about programs

- Does the program terminate?

Static analysis can be used to answer questions about programs

- Does the program terminate?
- Does the program divide by zero?

Static analysis can be used to answer questions about programs

- Does the program terminate?
- Does the program divide by zero?
- Is this variable used after assignment?

Static analysis can be used to answer questions about programs

- Does the program terminate?
- Does the program divide by zero?
- Is this variable used after assignment?
- How many conditions does the program have?

In answering interesting questions, static analysis abuts decidability and computability

In answering interesting questions, static analysis abuts decidability and computability

We make concessions

- Halting problem $\rightarrow$ Limit the depth of execution paths

In answering interesting questions, static analysis abuts decidability and computability

We make concessions

- Halting problem $\rightarrow$ Limit the depth of execution paths
- Decidability $\rightarrow$ Over-approximate values

In answering interesting questions, static analysis abuts decidability and computability

We make concessions

- Halting problem $\rightarrow$ Limit the depth of execution paths
- Decidability $\rightarrow$ Over-approximate values
- Soundness $\rightarrow$ Sometimes ditched for performance

Use different representations for different problems

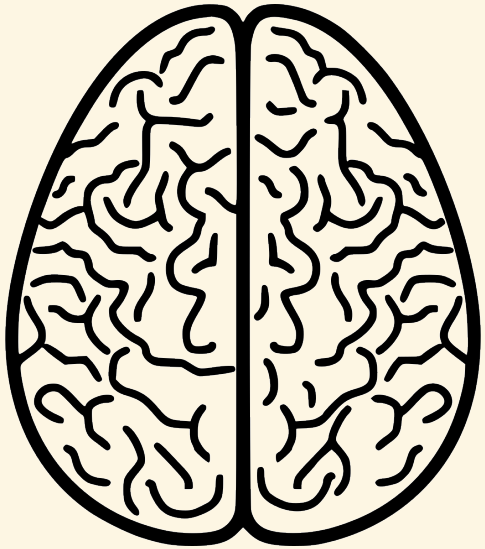
The internal interpreter

```
1   public int example(int n) {
2       int p0 = 0;
3       int p1 = 1;
4       int p = 0;
5
6       for (int i = 0; i < n; i++) {
7           p = p0 + p1;
8           p0 = p1;
9           p1 = p;
10      }
11
12      return p;
13  }
```

Use different representations for different problems

The internal interpreter

```
1   public int example(int n) {
2       int p0 = 0;
3       int p1 = 1;
4       int p = 0;
5
6       for (int i = 0; i < n; i++) {
7           p = p0 + p1;
8           p0 = p1;
9           p1 = p;
10      }
11
12      return p;
13  }
```



Use different representations for different problems

Textual Analysis

How many Lines of Code?

```
package hello;
import java.util.Optional;;
public class Example {
    /** return maybe an integer
     */
    public Optional<Integer> getNum() {
        /* int r = (new Random()).nextint();
           return Optional.of(r);
        */
        // nope, return nothing
        return Optional.empty();
    }
}
```

Use different representations for different problems

Textual Analysis

How many **source** Lines of Code (sloc)?

```
package hello;
import java.util.Optional;;
public class Example {
    /** return maybe an integer
     */
    public Optional<Integer> getNum() {
        /* int r = (new Random()).nextint();
           return Optional.of(r);
        */
        // nope, return nothing
        return Optional.empty();
    }
}
```

Use different representations for different problems

Abstract Syntax Trees

How many conditional branches?

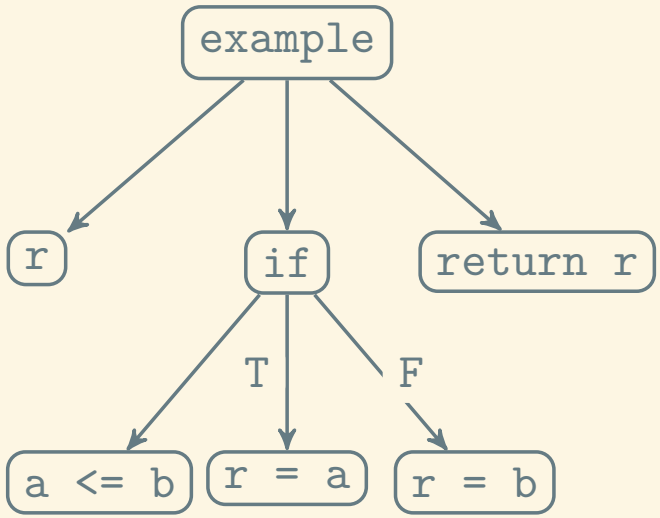
```
1   public int example(int a, int b) {
2       // if a > b then b
3       /* if a < b then a
4          if a == b then a */
5       int r = 0;
6       if (a <= b) {
7           r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```

Use different representations for different problems

Abstract Syntax Trees

How many conditional branches?

```
1   public int example(int a, int b) {
2       // if a > b then b
3       /* if a < b then a
4          if a == b then a */
5       int r = 0;
6       if (a <= b) {
7           r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```

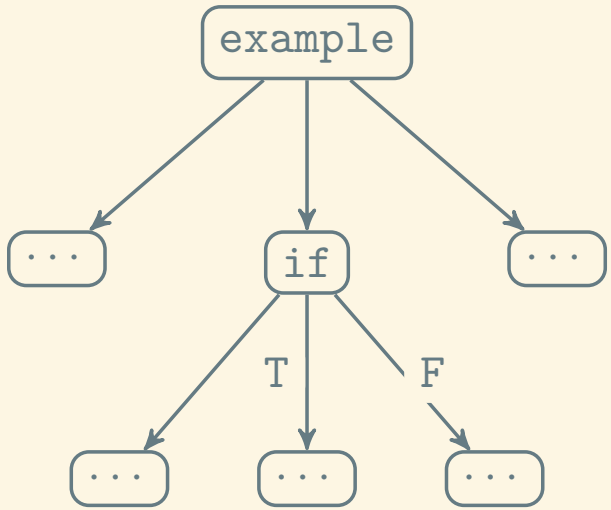


Use different representations for different problems

Abstract Syntax Trees

How many conditional branches?

```
1   public int example(int a, int b) {
2       // if a > b then b
3       /* if a < b then a
4          if a == b then a */
5       int r = 0;
6       if (a <= b) {
7           r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```



Use different representations for different problems

Control-Flow Graph

Does `r = 0` survive?

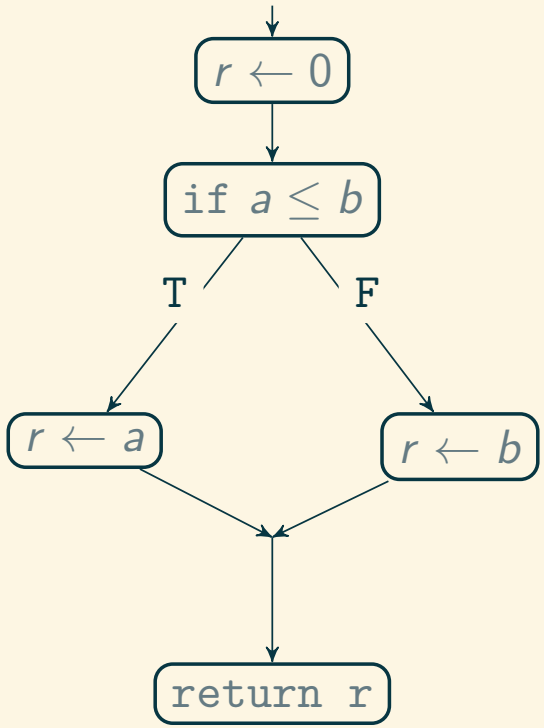
```
1   public int example(int a, int b) {
2       // if a > b then b
3       /* if a < b then a
4          if a == b then a */
5       int r = 0;
6       if (a <= b) {
7           r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```

Use different representations for different problems

Control-Flow Graph

Does `r = 0` survive?

```
1   public int example(int a, int b) {
2       // if a > b then b
3       /* if a < b then a
4          if a == b then a */
5       int r = 0;
6       if (a <= b) {
7           r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```



Static analysis is the process of computing facts

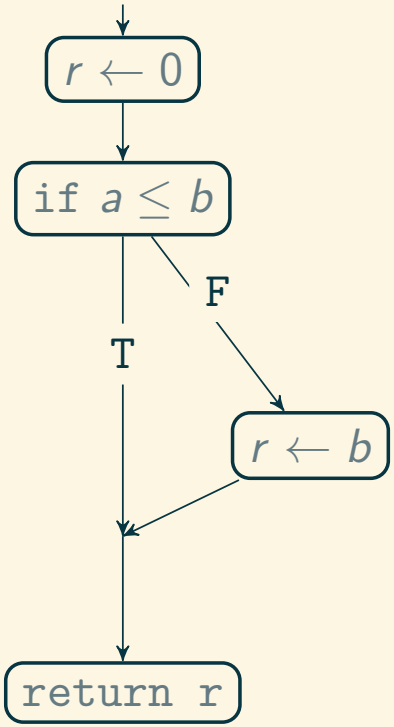
- Static analysis can be used to answer interesting questions
- Static analysis competes with many challenges such as computability and decidability
- Static analysis takes many forms

Defect finding through liveness

```
1   public int example(int a, int b) {
2       // if a > b then b
3       // if a < b then a
4       // if a == b then a
5       int r = 0;
6       if (a <= b) {
7           // r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```

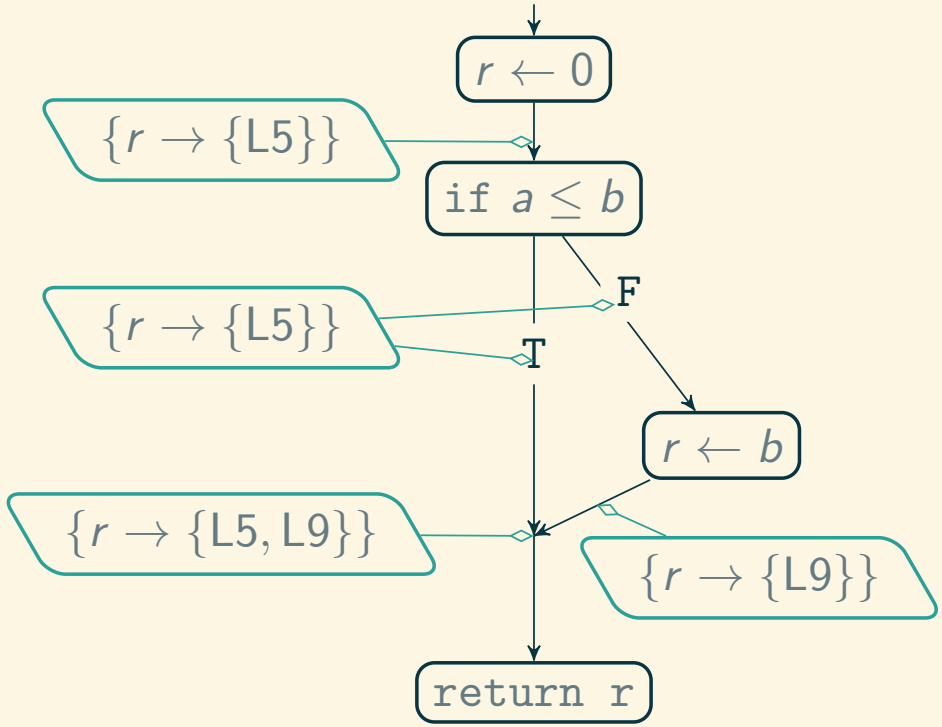
Defect finding through liveness

```
1   public int example(int a, int b) {
2       // if a > b then b
3       // if a < b then a
4       // if a == b then a
5       int r = 0;
6       if (a <= b) {
7           // r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```



Defect finding through liveness

```
1   public int example(int a, int b) {
2       // if a > b then b
3       // if a < b then a
4       // if a == b then a
5       int r = 0;
6       if (a <= b) {
7           // r = a;
8       } else {
9           r = b;
10      }
11      return r;
12  }
```



Program Optimization

Constant Propagation and Folding

```
1   int example() {
2       int x = 10;
3       int y = 30;
4       int z = x + x + x;
5       if (y == z) {
6           return 0;
7       } else {
8           return 1;
9       }
10  }
```

Program Optimization

Constant Propagation and Folding

```
1   int example() {
2       int x = 10;
3       int y = 30;
4       int z = x + x + x;
5       if (y == z) {
6           return 0;
7       } else {
8           return 1;
9       }
10  }
```

```
1   int example() {
2       int z = 10 + 10 + 10;
3       if (30 == z) {
4           return 0;
5       } else {
6           return 1;
7       }
8   }
```

Program Optimization

Constant Propagation and Folding

```
1   int example() {
2       int x = 10;
3       int y = 30;
4       int z = x + x + x;
5       if (y == z) {
6           return 0;
7       } else {
8           return 1;
9       }
10  }
```

```
1   int example() {
2       int z = 10 + 10 + 10;
3       if (30 == z) {
4           return 0;
5       } else {
6           return 1;
7       }
8   }
```

```
1   int example() {
2       if (30 == 30) {
3           return 0;
4       } else {
5           return 1;
6       }
7   }
```

Program Optimization

Constant Propagation and Folding

```
1   int example() {
2       int x = 10;
3       int y = 30;
4       int z = x + x + x;
5       if (y == z) {
6           return 0;
7       } else {
8           return 1;
9       }
10  }
```

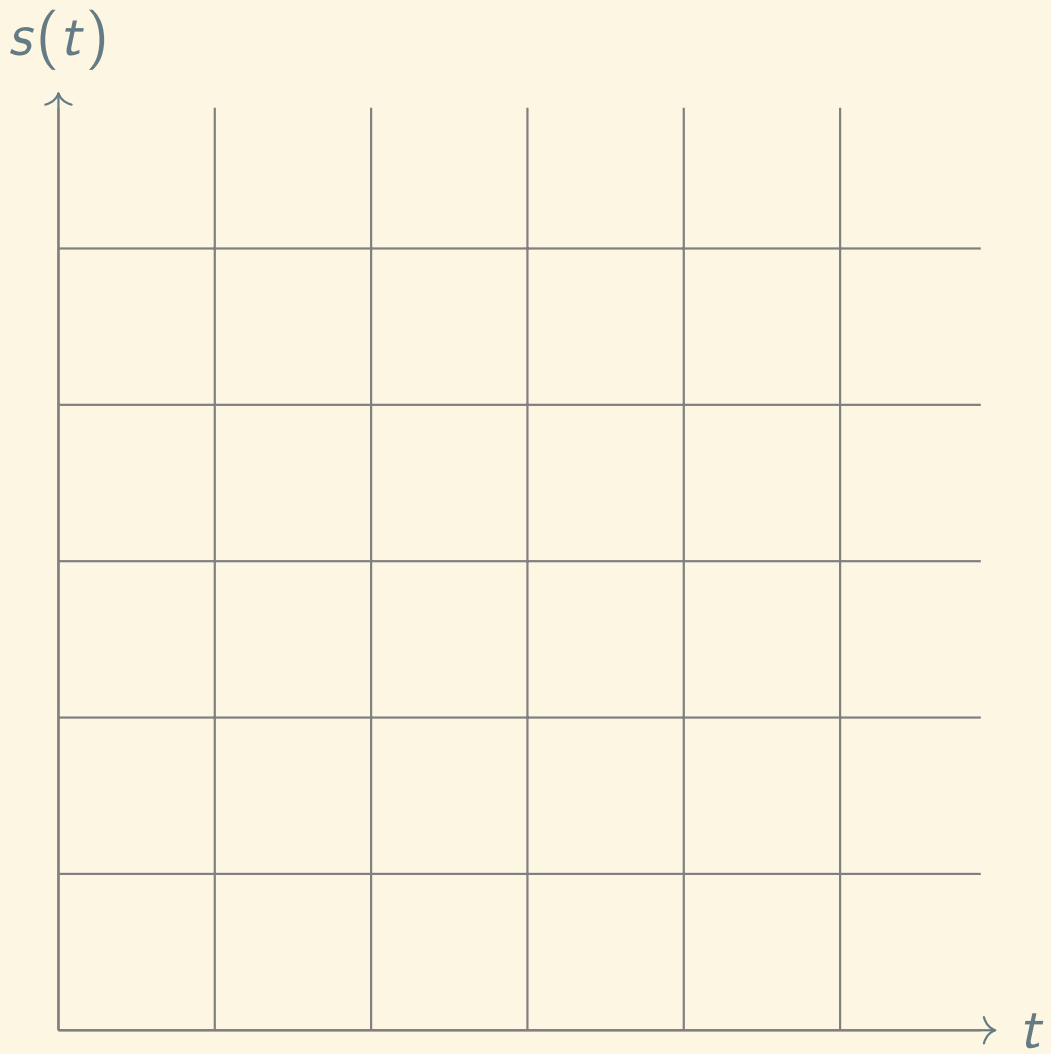
```
1   int example() {
2       int z = 10 + 10 + 10;
3       if (30 == z) {
4           return 0;
5       } else {
6           return 1;
7       }
8   }
```

```
1   int example() {
2       if (30 == 30) {
3           return 0;
4       } else {
5           return 1;
6       }
7   }
```

```
int example() {
    return 0;
}
```

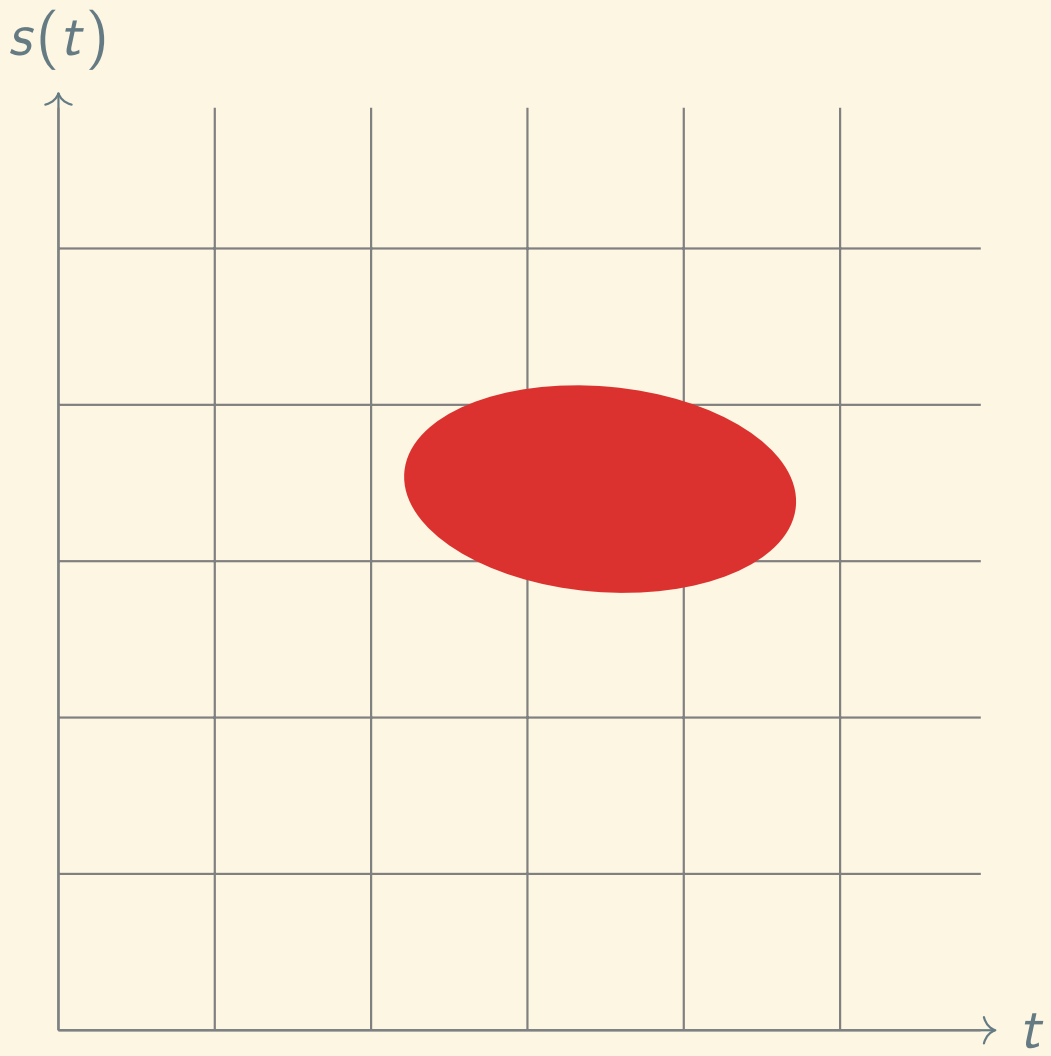
Abstract Interpretation

Big Picture [2]



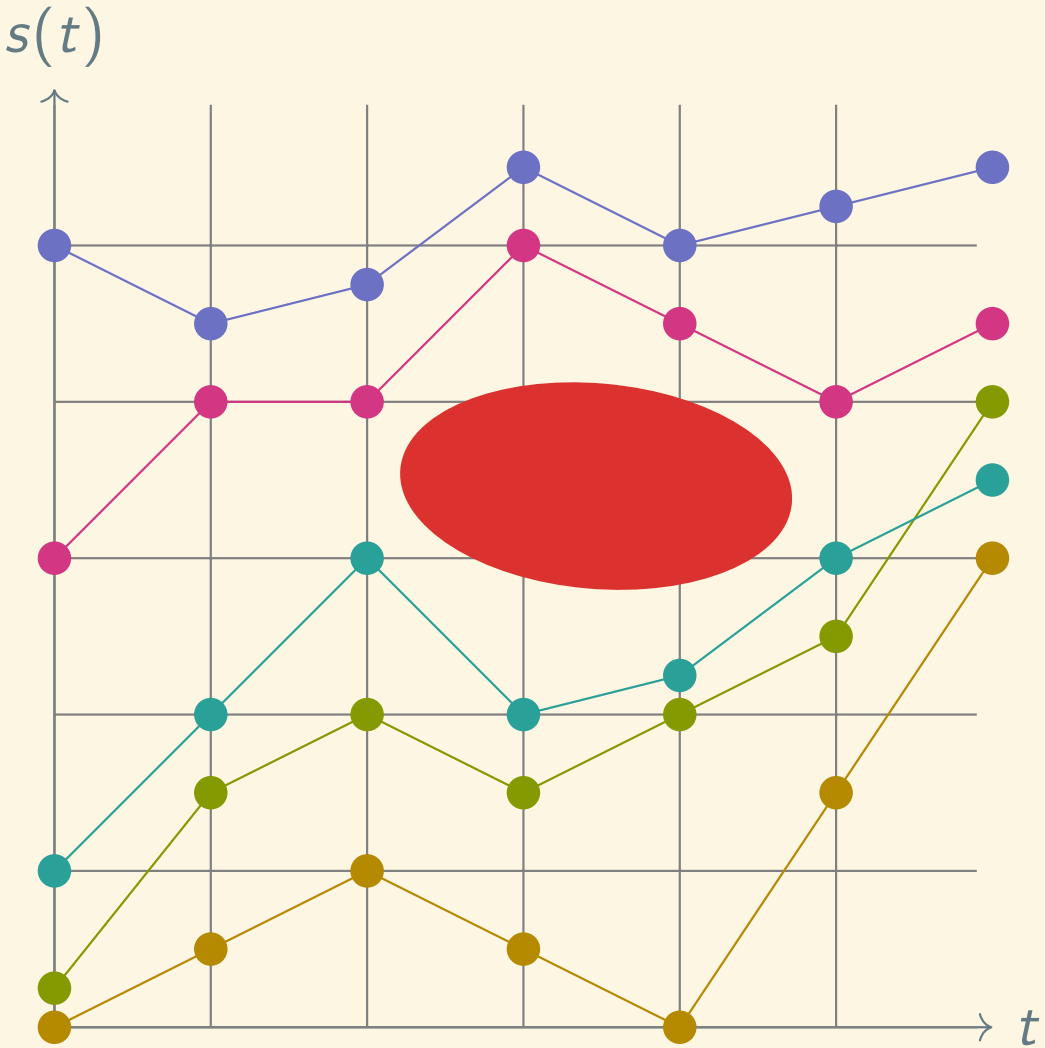
Abstract Interpretation

Big Picture [2]



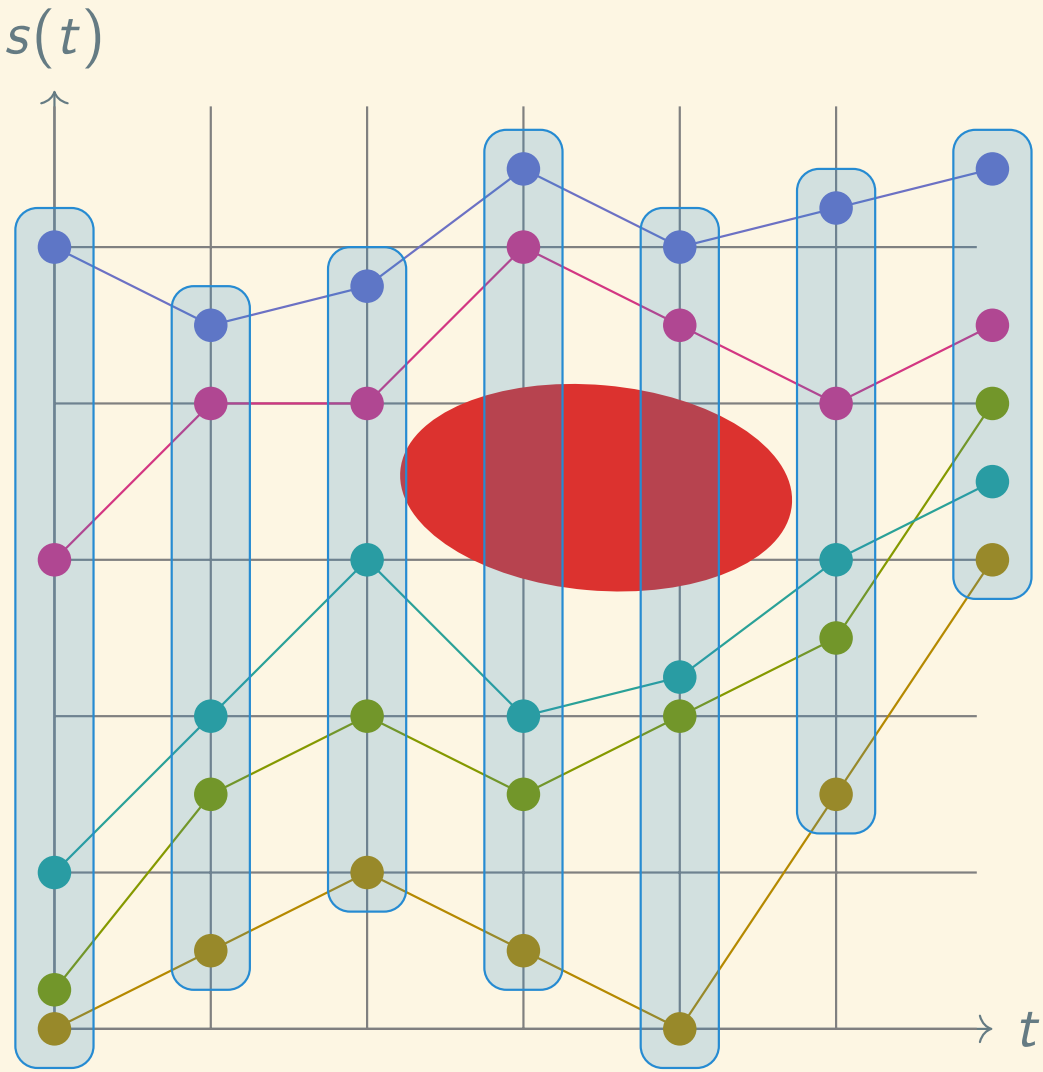
Abstract Interpretation

Big Picture [2]



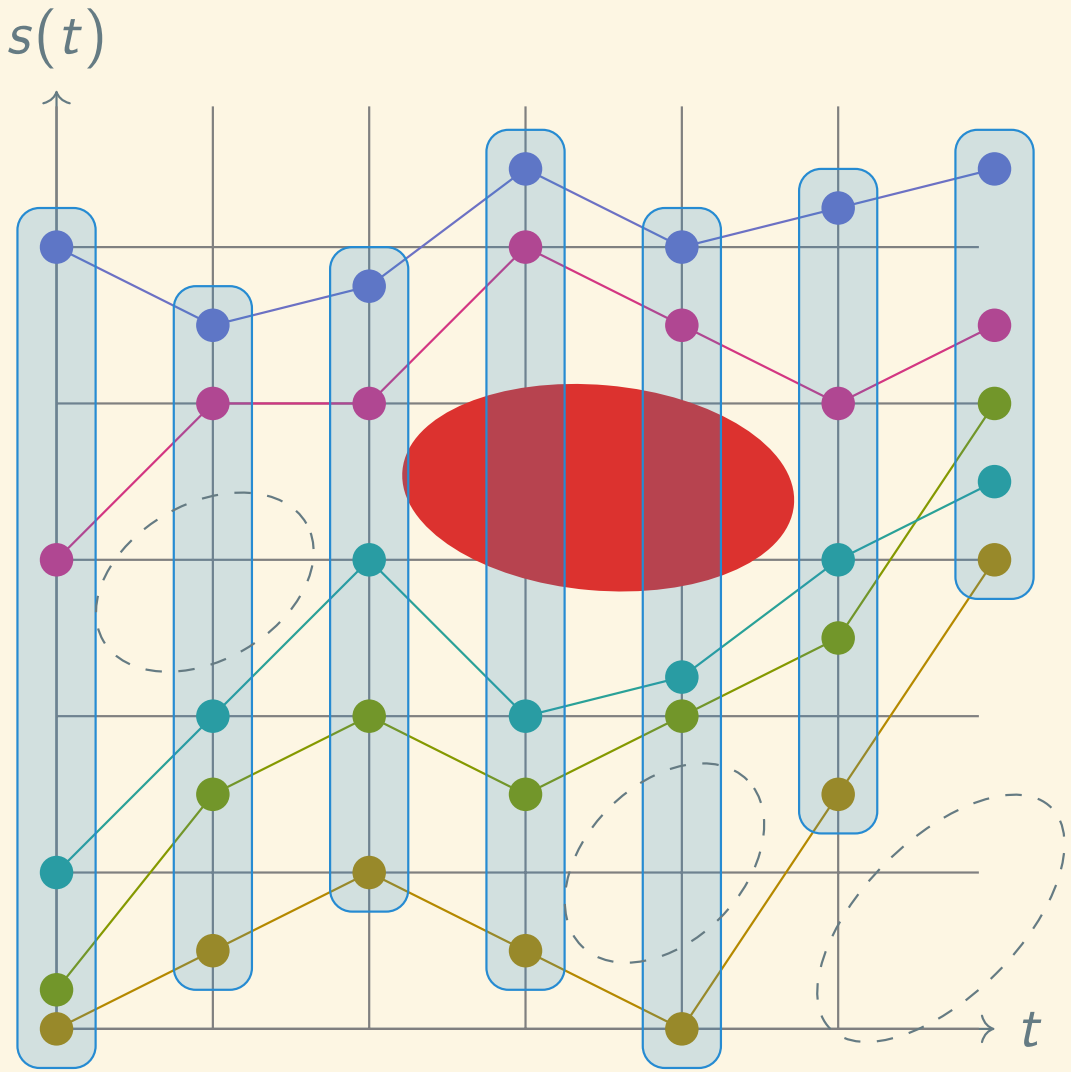
Abstract Interpretation

Big Picture [2]

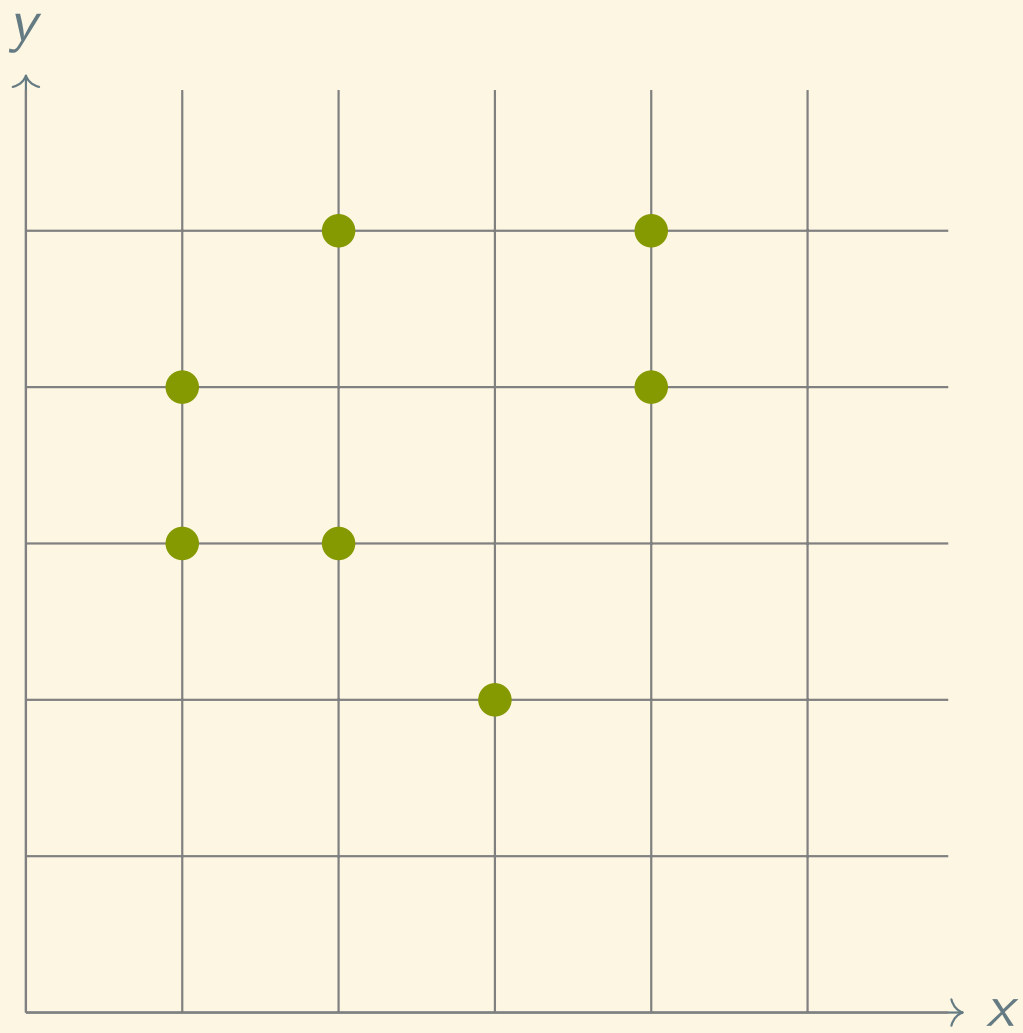


Abstract Interpretation

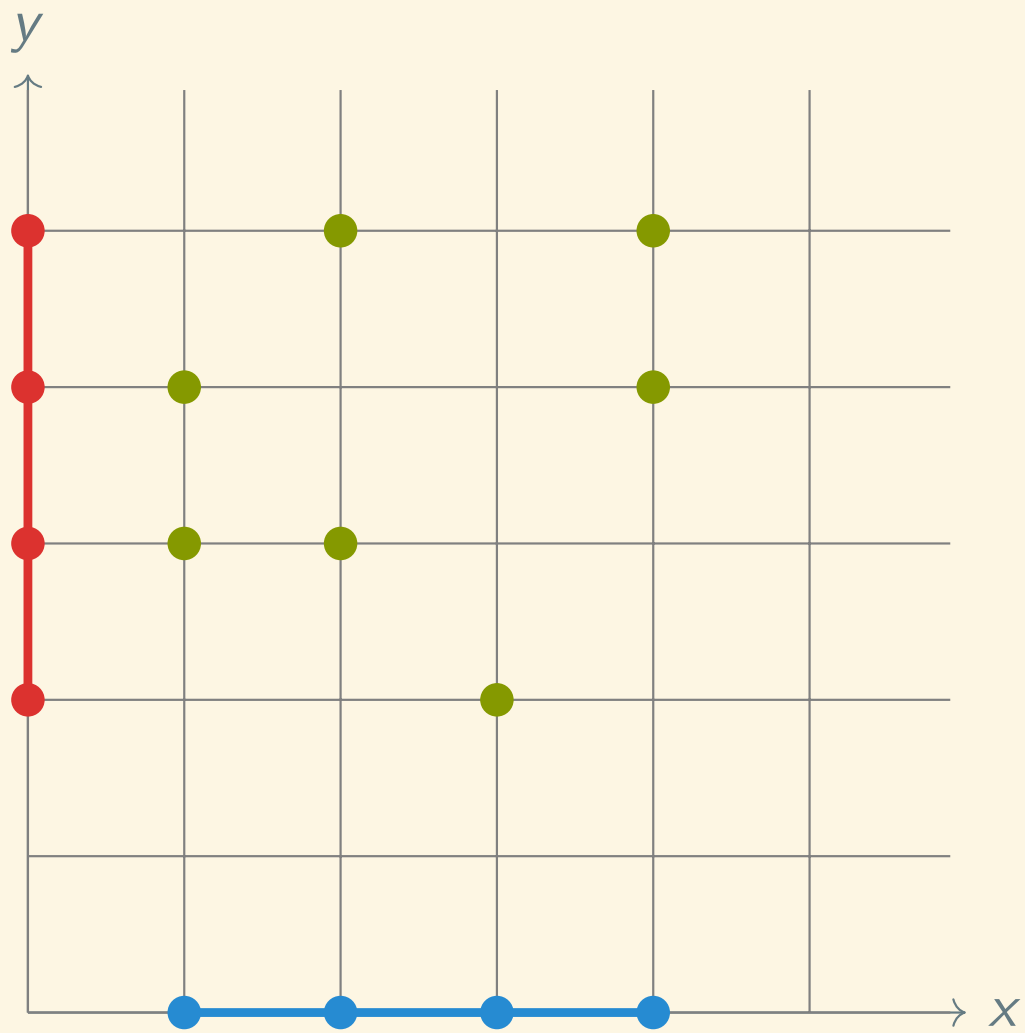
Big Picture [2]



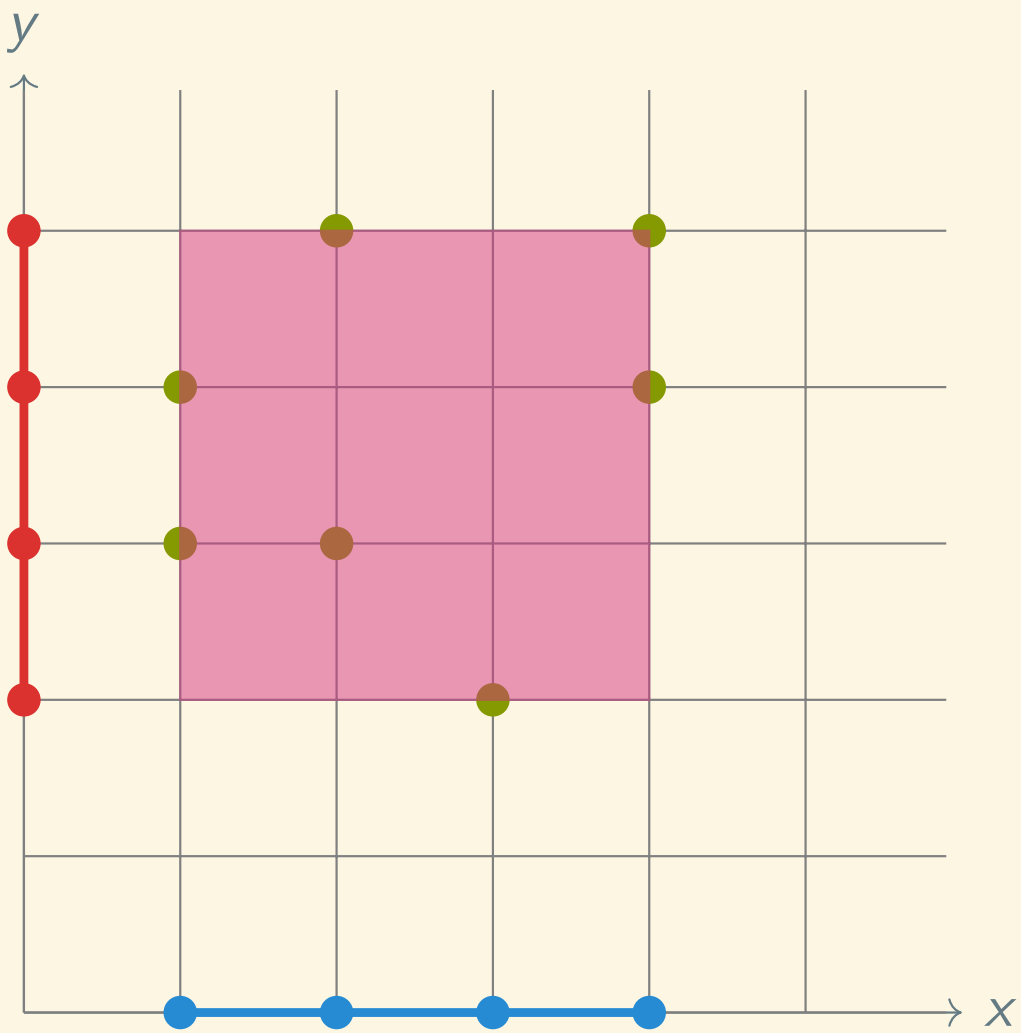
Examples of Precision



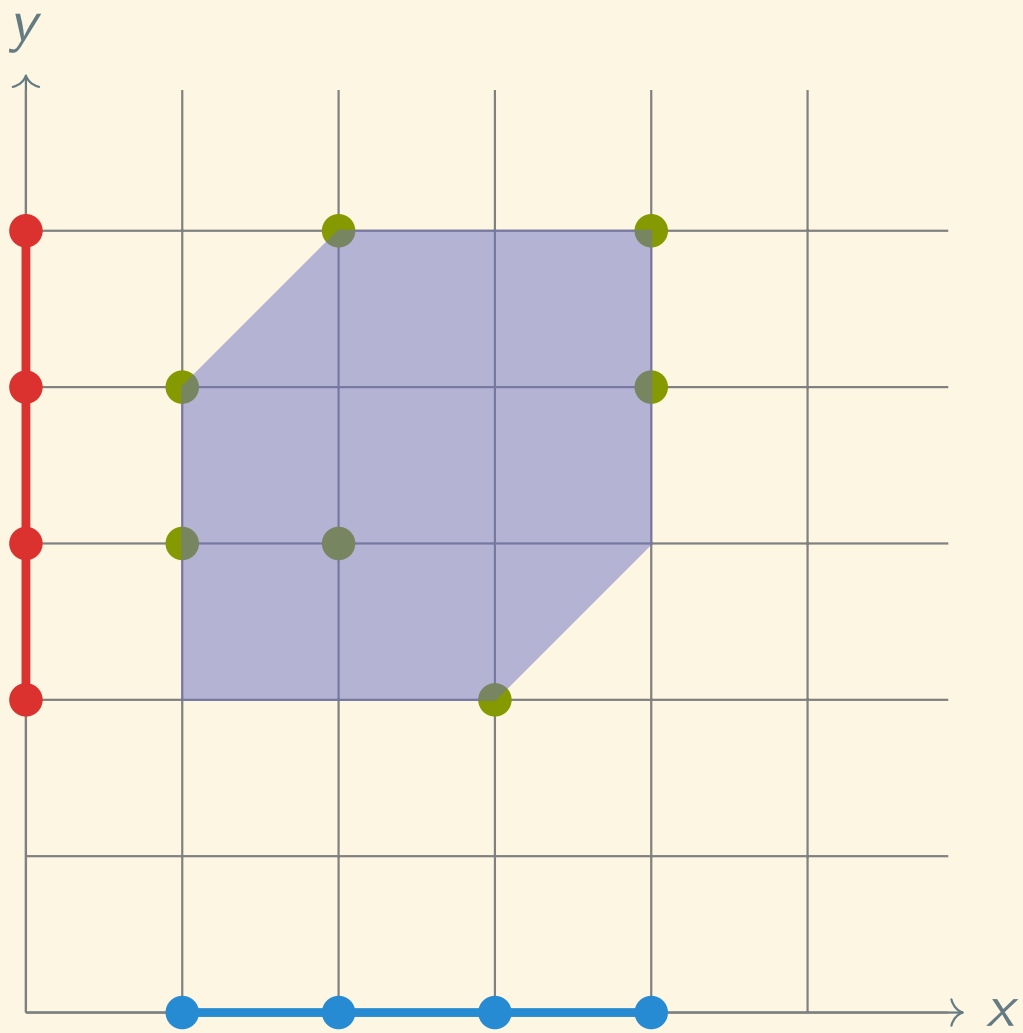
Examples of Precision



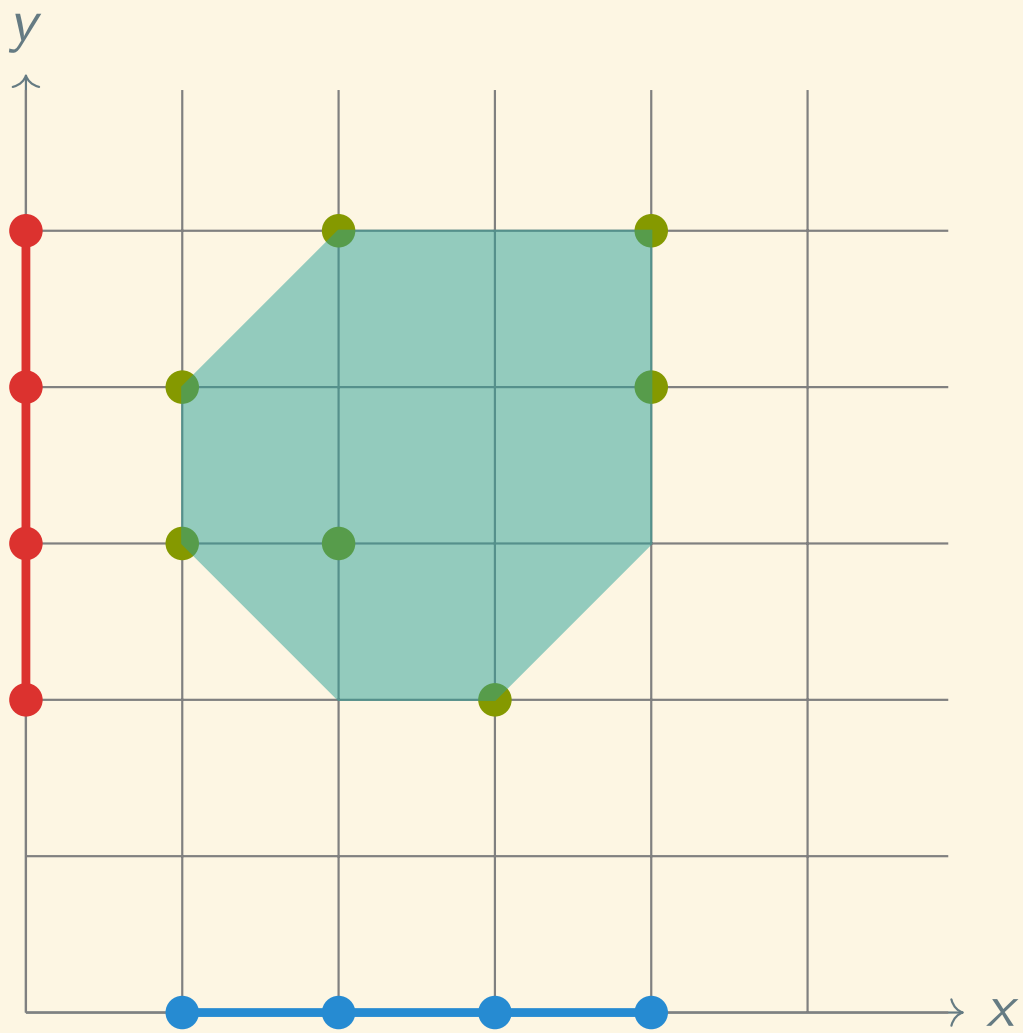
Examples of Precision



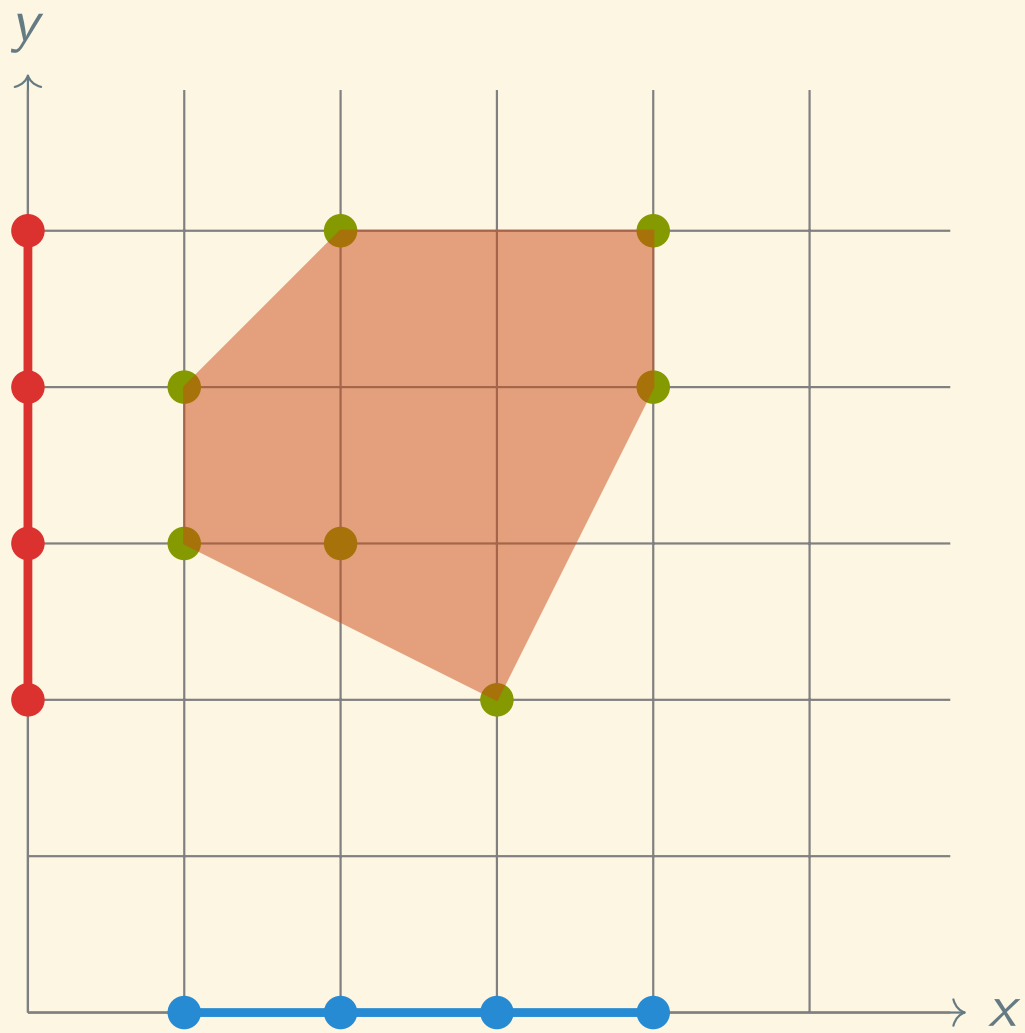
Examples of Precision



Examples of Precision



Examples of Precision



Identifying Minimal Changes in the Zone Abstract Domain

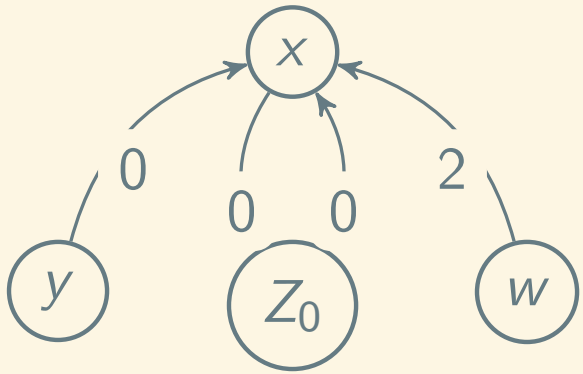
Zones Representation

$$x = 0$$
$$w - x \leq 2$$
$$y - x \leq 0$$

Identifying Minimal Changes in the Zone Abstract Domain

Zones Representation

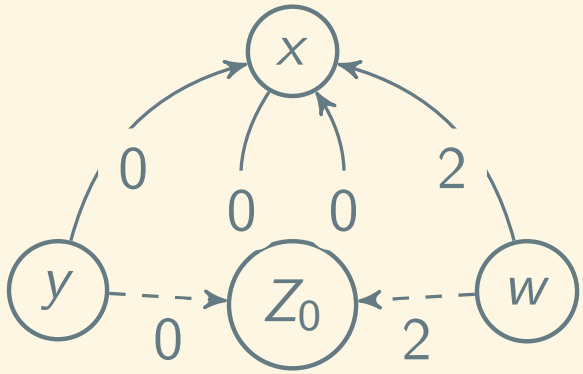
$$x = 0$$
$$w - x \leq 2$$
$$y - x \leq 0$$



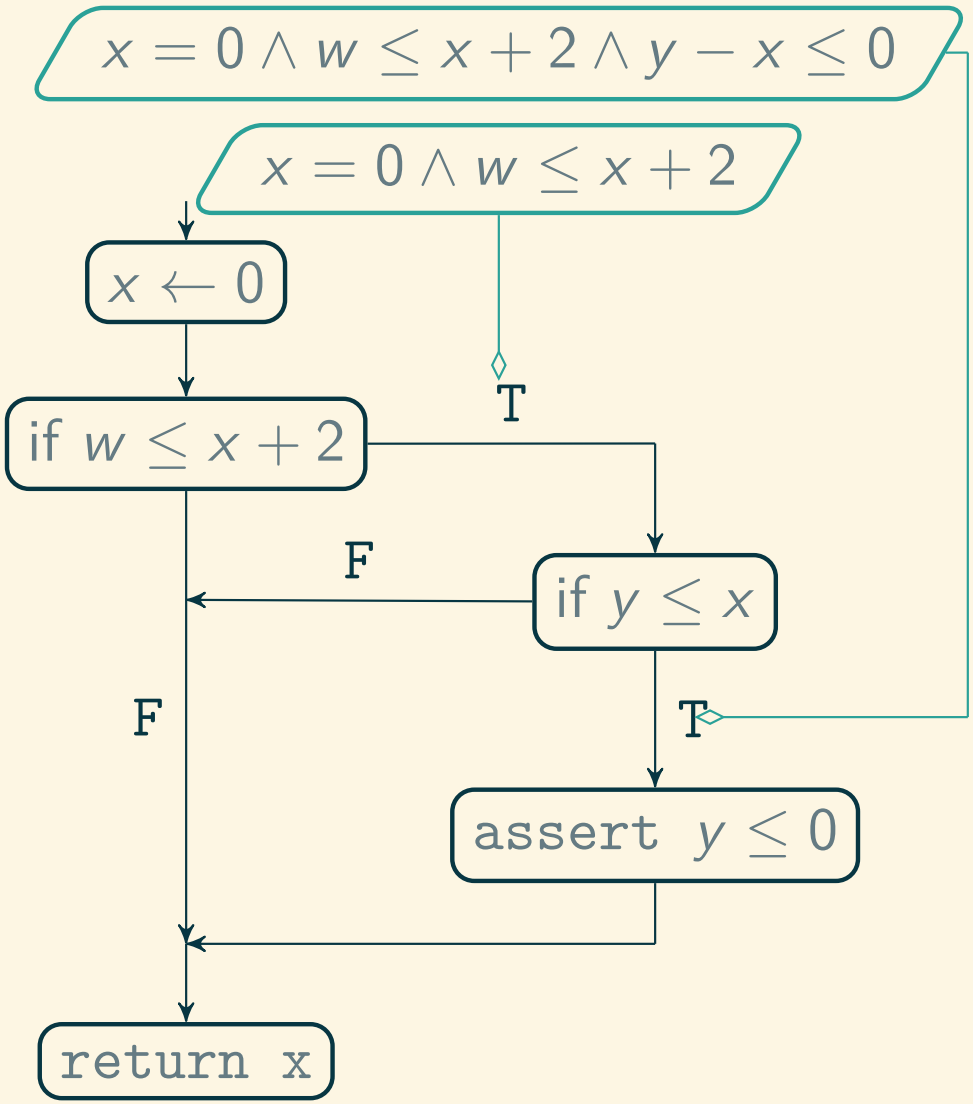
Identifying Minimal Changes in the Zone Abstract Domain

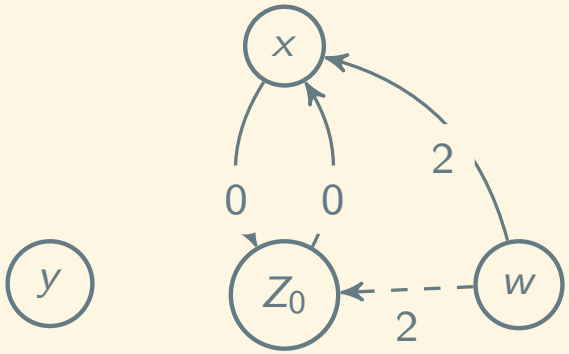
Zones Representation

$$x - Z_0 = 0$$
$$w - x \leq 2$$
$$y - x \leq x$$

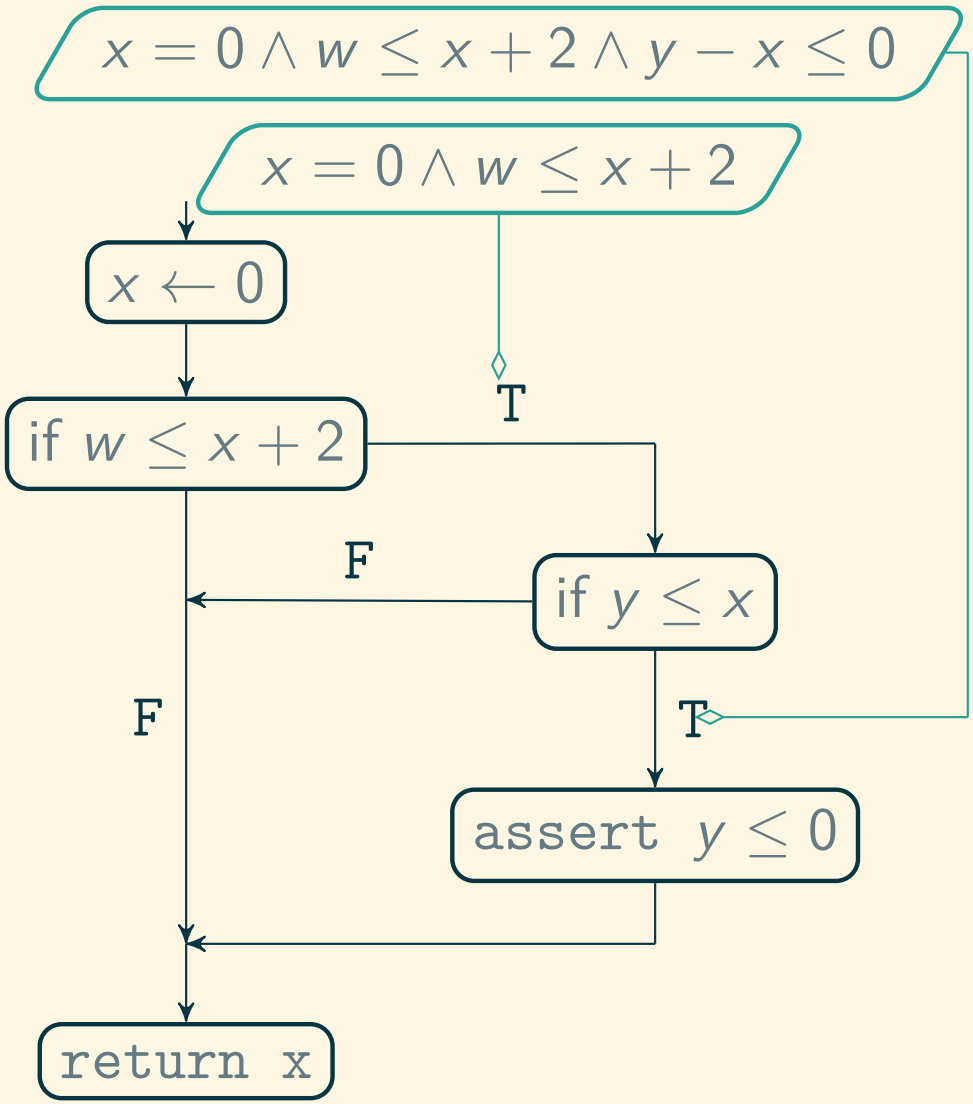


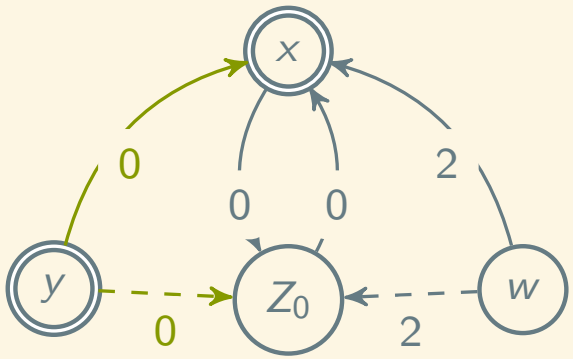
Identifying Minimal Changes in the Zone Abstract Domain





Identifying Minimal Changes in the Zone Abstract Domain





Discussion

- Static analysis computes facts about programs.
- There are many uses for static analysis.
  - "Linting"
  - Defect Detection
  - Program Optimization
  - Formal Verification
- Static analysis is not without its challenges
  - Computability
  - Decidability

Thank you

Questions?

Some of the work reported here was supported by the U.S. National Science Foundation under award CCF-19-42044.

References I

[1] Kenny Ballou and Elena Sherman. *Identifying Minimal Changes in the Zone Abstract Domain*. 2023. DOI: 10.48550/ARXIV.2304.14550. URL: https://arxiv.org/abs/2304.14550.

[2] Patrick Cousot. *Patrick Cousot's slides of talks on abstract interpretation*. July 2020. URL: https://pcousot.github.io/talks.html (visited on 06/25/2023).

[3] Flemming Nielson, Hanne R. Nielson, and Chris Hankin. *Principles of Program Analysis*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999. ISBN: 9783540654100.

Abstraction and Concretization

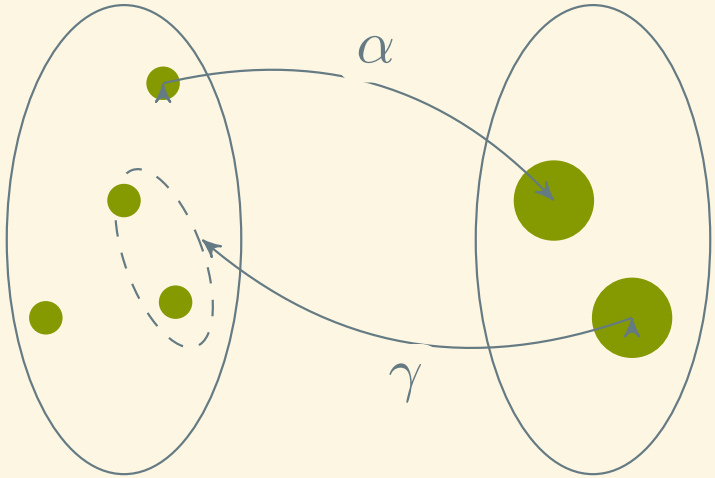


Figure: Image inspired from similar images in *Principles of Program Analysis*