Detecting Anomalies in the Order of Equally-typed Method Arguments

Michael Pradel, Thomas R. Gross

Department of Computer Science
ETH Zurich

Image to have two different keys...

Image to have two different keys...

Which key fits which keyhole?

```
int high = ...
int low = ...
```

Image to have two `int` variables...

```
void setEndPoints(int, int)
```

```
int high = ...
int low = ...
```

Image to have two `int` variables...

```
void setEndPoints(int, int)
```

Which argument fits which position?

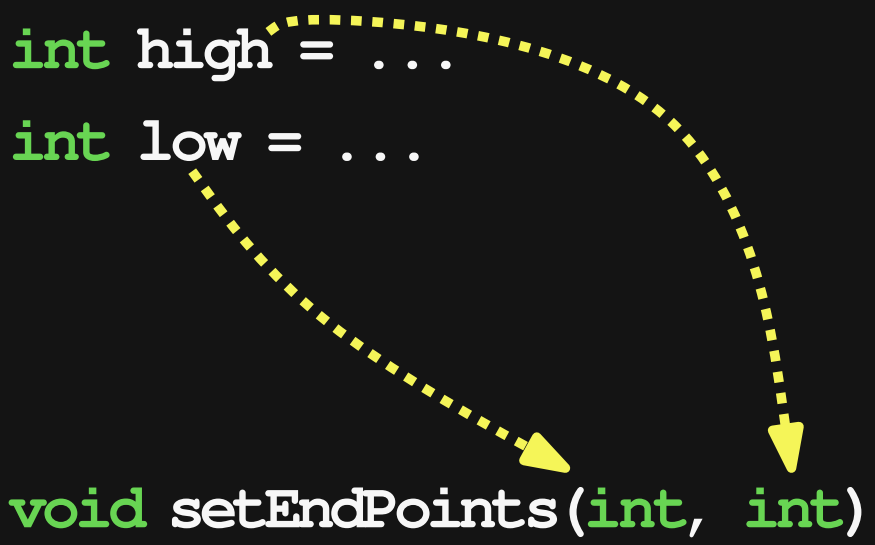


Image to have two `int` variables...

Which argument fits which position?

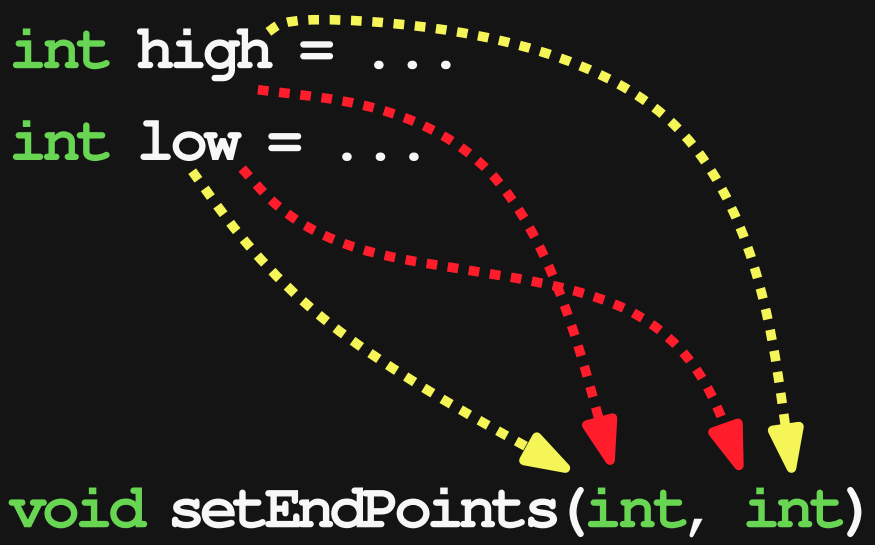


Image to have two `int` variables...

Which argument fits which position?

Equally-typed Method Arguments

Problem: Type system doesn’t help

What Can Go Wrong? (1)

```
int high = ..;
int low = ..;

setEndPoints(?, ?);

void setEndPoints(int i, int j) {
   ...
}
```

What Can Go Wrong? (1)

```
int high = ..;
int low = ..;

setEndPoints(?, ?);

void setEndPoints(int i, int j) {
   ...
}
```

Bad names for formal parameters

→ Understandability problem

What Can Go Wrong? (2)

```
int high = ..;
int low = ..;

setEndPoints(low, high);

void setEndPoints(int high, int low) {
   ...
}
```

What Can Go Wrong? (2)

```
int high = ..;
int low = ..;

setEndPoints(low, high);

void setEndPoints(int high, int low) {
   ...
}
```

Arguments passed in wrong order

→ Correctness problem

What Can Go Wrong? (3)

```
int high = ..;
int low = ..;

setEndPoints(low, high);  // invert end points

void setEndPoints(int high, int low) {
   ...
}
```

What Can Go Wrong? (3)

```
int high = ..;
int low = ..;

setEndPoints(low, high);  // invert end points

void setEndPoints(int high, int low) {
   ...
}
```

Unexpected but correct argument order

$\rightarrow$ **Maintainability problem**

Is This Important?

11% of all call sites have two
or more equally-typed arguments

I.e., 1 unchecked call per 20 LOC

(DaCapo benchmarks, 1.6 MLOC Java code)

Static Anomaly Detection

Program → Argument Naming Examples → Anomalies

```
void m(int a,
       int b) {}
m(a, b);
m(b, a);
```

(a,b)
(a,b)
(b,a)

```
m(b, a);
```

Reverse?!

Static Anomaly Detection

Simple yet effective

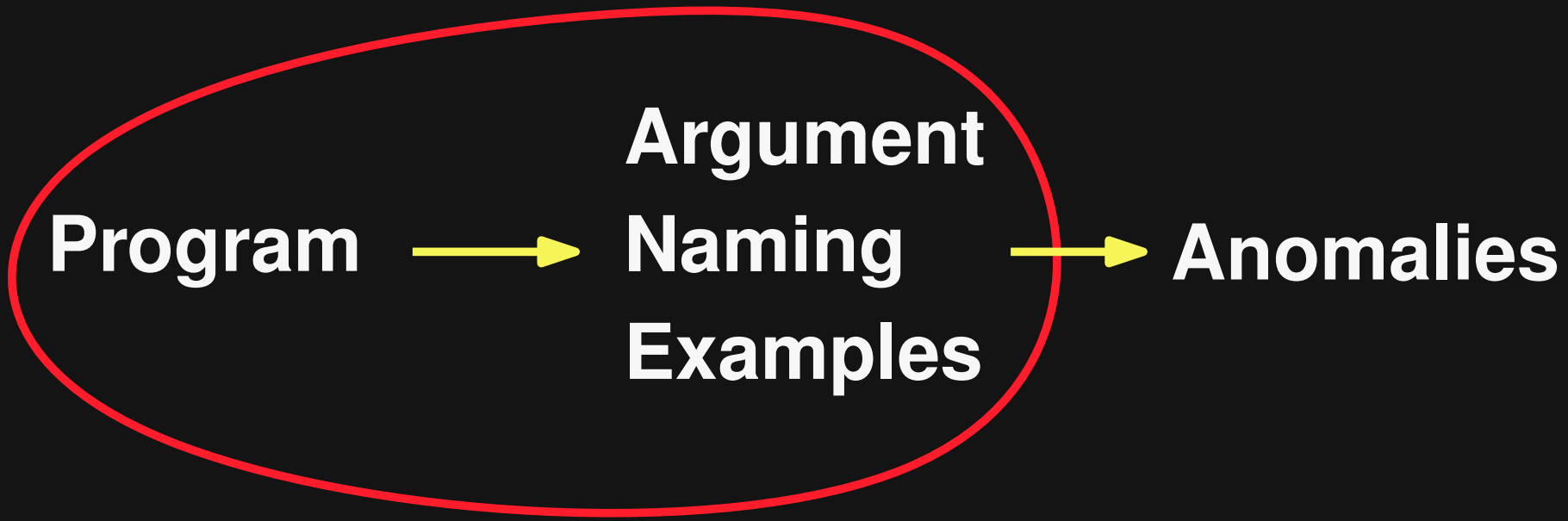
```
void m(int a,
       int b) {}
m(a, b);
m(b, a);
```

(a,b)
(a,b)
(b,a)

```
m(b, a);
```

Reverse?!

Static Anomaly Detection




```
void m(int a,
       int b) {}
m(a, b);
m(b, a);
```

(a,b)
(a,b)
(b,a)

```
m(b, a);
```

Reverse?!

Argument Name Extraction

Goal: Find examples for argument names

```
int highEP;
int[] lowEP;
Data data;

setEndPoints(highEP, lowEP[i]);

setEndPoints(data.h, data.low());
```

Argument Name Extraction

Goal: Find examples for argument names

```
int highEP;
int[] lowEP;
Data data;

setEndPoints(highEP, lowEP[i]);

setEndPoints(data.h, data.low());
```

local variable: ”highEP”

Argument Name Extraction

Goal: Find examples for argument names

```
int highEP;
int[] lowEP;
Data data;

setEndPoints(highEP, lowEP[i]);

setEndPoints(data.h, data.low());
```

local variable: "highEP"

array access: "lowEP"

Argument Name Extraction

Goal: Find examples for argument names

```
int highEP;
int[] lowEP;
Data data;

setEndPoints(highEP, lowEP[i]);

setEndPoints(data.h, data.low());
```

local variable: ”highEP”

array access: ”lowEP”

field access: ”h”

Argument Name Extraction

Goal: Find examples for argument names

```
int highEP;
int[] lowEP;
Data data;

setEndPoints(highEP, lowEP[i]);

setEndPoints(data.h, data.low());
```

local variable: ”highEP”

array access: ”lowEP”

field access: ”h”

method call: ”low”

Parameter Name Extraction

More examples: Formal parameter names

```
void setEndPoints(int high, int low) {
  ...
}
```

Static Anomaly Detection

Program → Argument Naming Examples → Anomalies

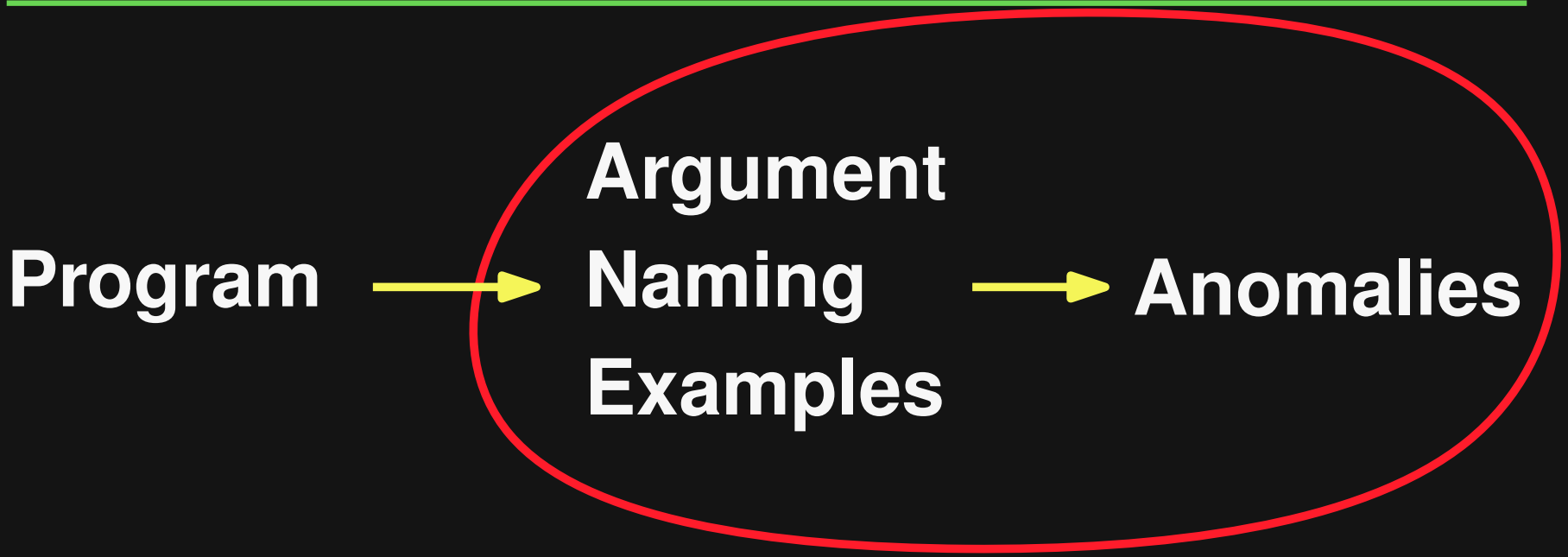
```
void m(int a,
       int b) {}
m(a, b);
m(b, a);
```

(a,b)
(a,b)
(b,a)

```
m(b, a);
```

Reverse?!

Static Anomaly Detection




```
void m(int a,
       int b) {}
m(a, b);
m(b, a);
```

(a,b)
(a,b)
(b,a)

```
m(b, a);
```

Reverse?!

Anomaly Detection

What is an anomaly?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
low	high

Anomaly Detection

What is an anomaly?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
low	high

An unusual name is no anomaly

Anomaly Detection

What is an anomaly?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
low	high

An unusual permutation is an anomaly

Anomaly Detection

Which permutation is more normal?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
low	high

vs.

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
high	low

Anomaly Detection

How normal is this permutation?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
high	low

Anomaly Detection

How normal is this permutation?

$$fitPos(\textbf{high}, 1) = similarity(\textbf{high}, 1) - similarity(\textbf{high}, 2)$$

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
high	low

Anomaly Detection

How normal is this permutation?

$$fitPos(\textbf{high}, 1) = similarity(\textbf{high}, 1) - similarity(\textbf{high}, 2)$$

$$normality = fitPos(\textbf{high}, 1) + fitPos(\textbf{low}, 2) - fitPos(\textbf{low}, 1) - fitPos(\textbf{high}, 2)$$

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
high	low

Anomaly Detection

Which permutation is more normal?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
low	high

$normality = 0\%$

vs.

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
high	low

$normality = 86\%$

Anomaly Detection

Which permutation is more normal?

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
low	high

$normality = 0\%$

vs.

Anomaly!

Pos. 1	Pos. 2
high	low
h	Low
high	low
highEP	lowEP
high	low

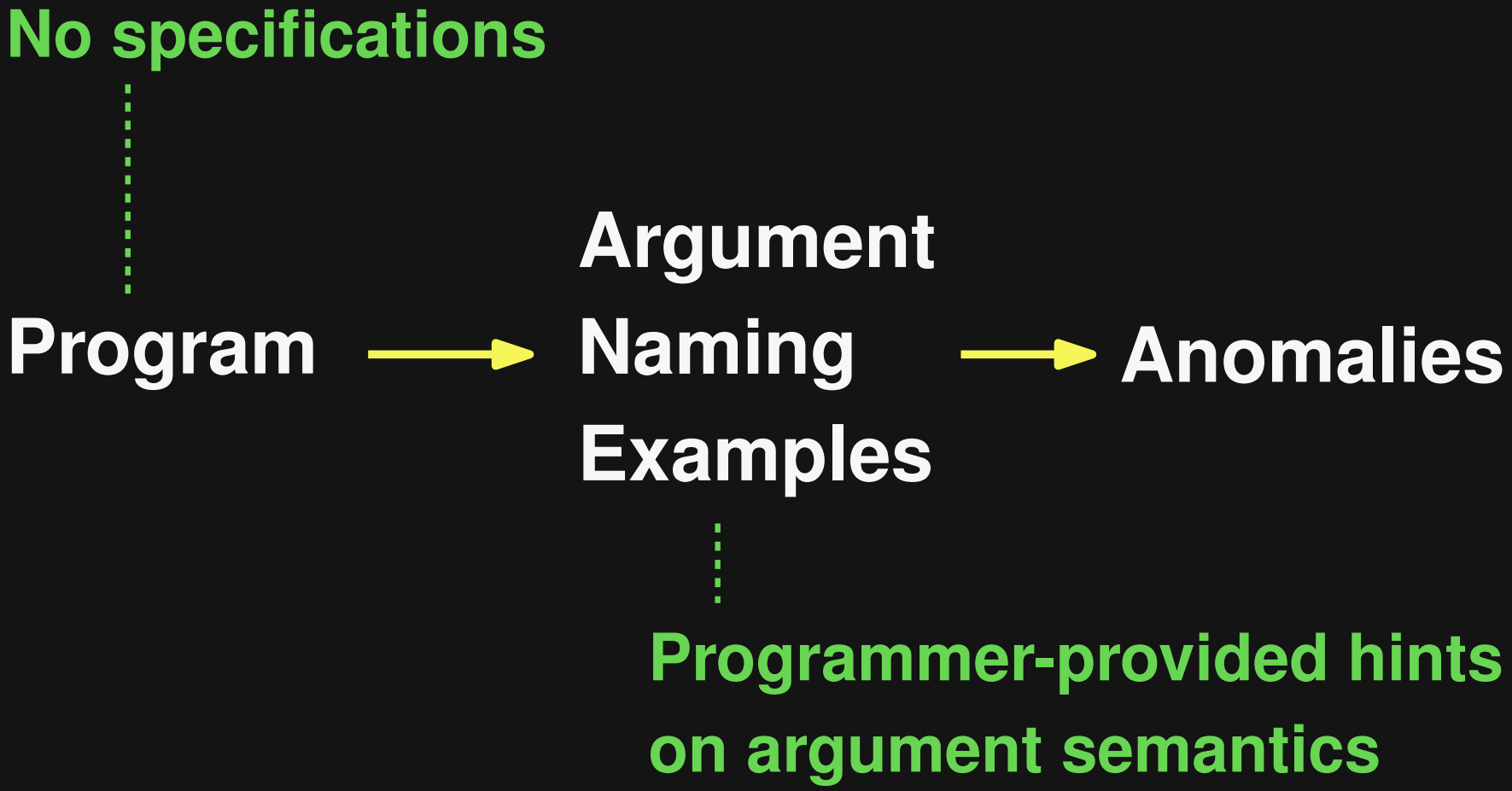
$normality = 86\%$

Summary of Approach

Summary of Approach

No specifications

Summary of Approach



Evaluation

- Effectiveness in finding anomalies?
- Anomalies in mature programs?

Setup:

DaCapo benchmarks, 1.6 MLOC Java

Effectiveness

for each call site with equally-typed arguments
    for each permutation of the arguments
        seed anomaly and try to find it

Effectiveness

for each call site with equally-typed arguments
    for each permutation of the arguments
        seed anomaly and try to find it

$$\text{Precision} = \frac{\text{\# true pos.}}{\text{\# true pos.} \;+\; \text{\# false pos.}}$$

$$\text{Recall} = \begin{cases} 1 & \text{if seeded anomaly found} \\ 0 & \text{otherwise} \end{cases}$$

Effectiveness

Average over 49K seeded anomalies

Precision: 72%

Recall: 38%

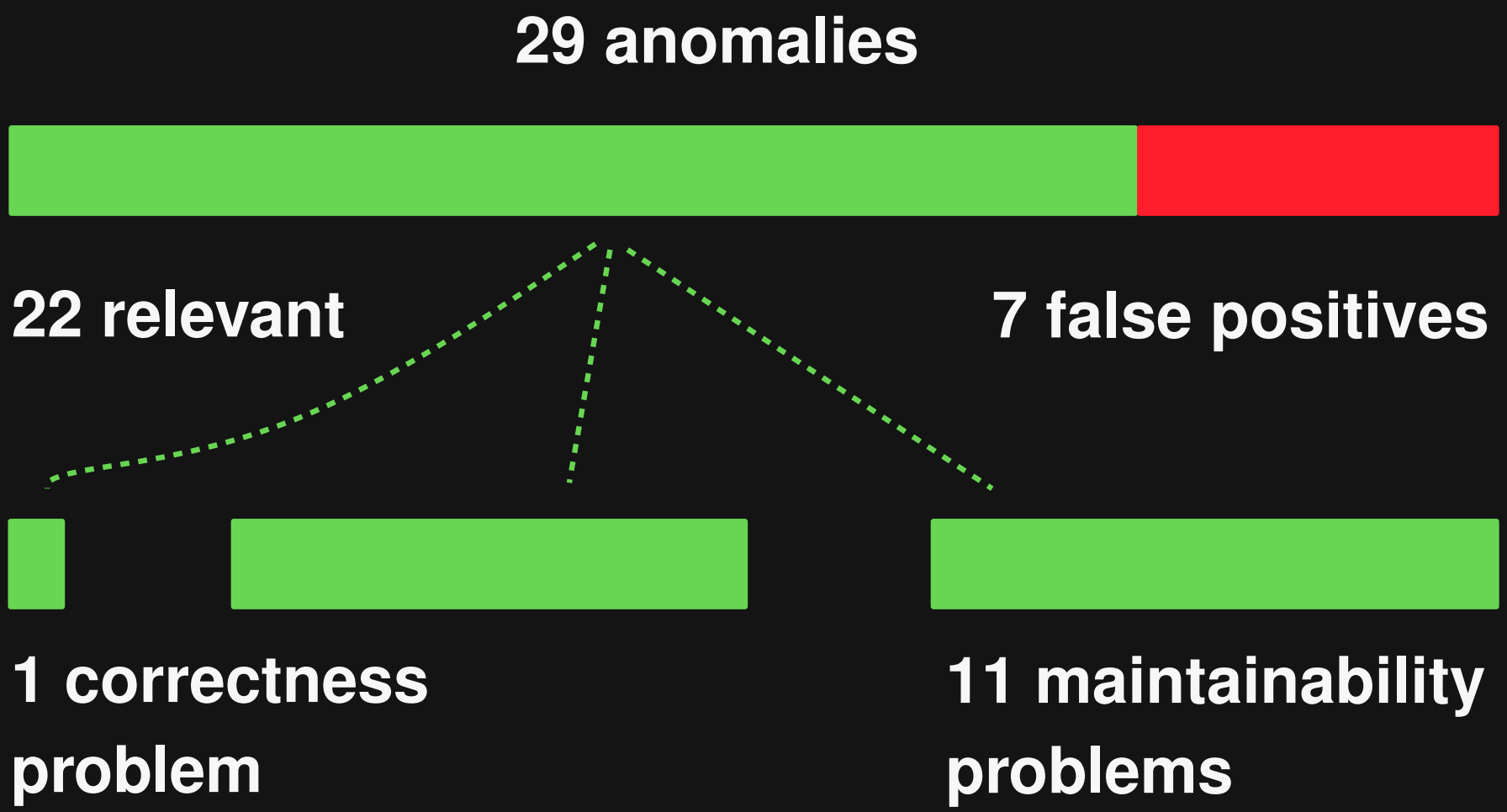
Anomalies in Real Programs

29 anomalies

22 relevant

7 false positives

Anomalies in Real Programs



Anomalies in Real Programs

Correctness problem in Eclipse

```
// call
createAlignment(name, mode,
   Alignment.R_INNERMOST, count, sourceRestart,
   adjust);

// called method
Alignment createAlignment(String name, int mode,
   int count, int sourceRestart, int continuationIndent,
   boolean adjust) { ... }
```

Anomalies in Real Programs

Correctness problem in Eclipse

```
// call
createAlignment(name, mode,
   Alignment.R_INNERMOST, count, sourceRestart,
   adjust);

// called method
Alignment createAlignment(String name, int mode,
   int count, int sourceRestart, int continuationIndent,
   boolean adjust) { ... }
```

Anomalies in Real Programs

Correctness problem in Eclipse

```
// call
createAlignment(name, mode,
   Alignment.R_INNERMOST, count, sourceRestart,
   adjust);

// called method
Alignment createAlignment(String name, int mode,
   int count, int sourceRestart, int continuationIndent,
   boolean adjust) { ... }
```

Anomalies in Real Programs

Understandability problem in Jython

```
// called method
PyFloat _pow(double value, double iw, PyObject modulo)
```

Anomalies in Real Programs

Understandability problem in Jython

```
// called method
PyFloat _pow(double value, double iw, PyObject modulo)
```

Exponentiation: What is base and what is exponent?

Anomalies in Real Programs

Understandability problem in Jython

```
// call
_pow(coerce(left), value, null)
```


```
// called method
PyFloat _pow(double value, double iw, PyObject modulo)
```

Exponentiation: What is base and what is exponent?

Anomalies in Real Programs

Maintainability problem in Eclipse

```
// call
generateOptimizedBoolean(
   currentScope, codeStream,
   falseLabel, trueLabel, valueRequired)

// called method
void generateOptimizedBoolean(
   BlockScope currentScope, CodeStream codeStream,
   Label trueLabel, Label falseLabel, boolean valueRequired)
```

Anomalies in Real Programs

Maintainability problem in Eclipse

```
// call
generateOptimizedBoolean(
   currentScope, codeStream,
   falseLabel, trueLabel, valueRequired)

// called method
void generateOptimizedBoolean(
   BlockScope currentScope, CodeStream codeStream,
   Label trueLabel, Label falseLabel, boolean valueRequired)
```

Unexpected but correct

Summary of Evaluation

- **Effectiveness in finding anomalies?**

  72% precision, 38% recall

- **Anomalies in mature programs?**

  22 relevant among 29 reported

What Does It Cost?

Only input:
Source code of program

Time to analyze 1.6 MLOC:
Less than two minutes

Conclusion

- Cheap technique to find anomalies involving **equally-typed arguments**
- **Simple** but **effective**
- Try it out!
  http://mp.binaervarianz.de/issta2011

Thank you!

michael@binaervarianz.de

http://mp.binaervarianz.de/issta2011