Automatic Testing of Sequential and Concurrent Substitutability

Michael Pradel and Thomas R. Gross

Department of Computer Science
ETH Zurich

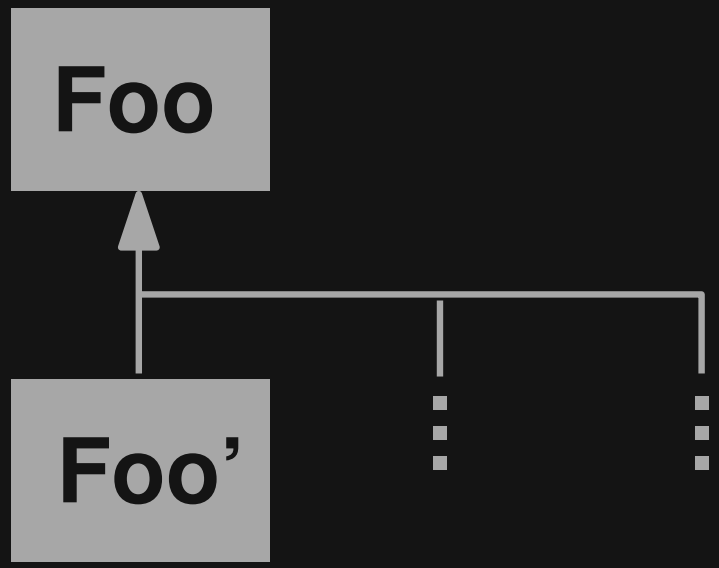
Motivation

```
void bar(Foo f) {
  f.m();
  ...
}
```

bar() expects functionality from Foo.m()

Motivation

```
void bar(Foo f) {
   f.m();
   ...
}
```



bar() expects functionality from Foo.m()

... even if Foo has subclasses

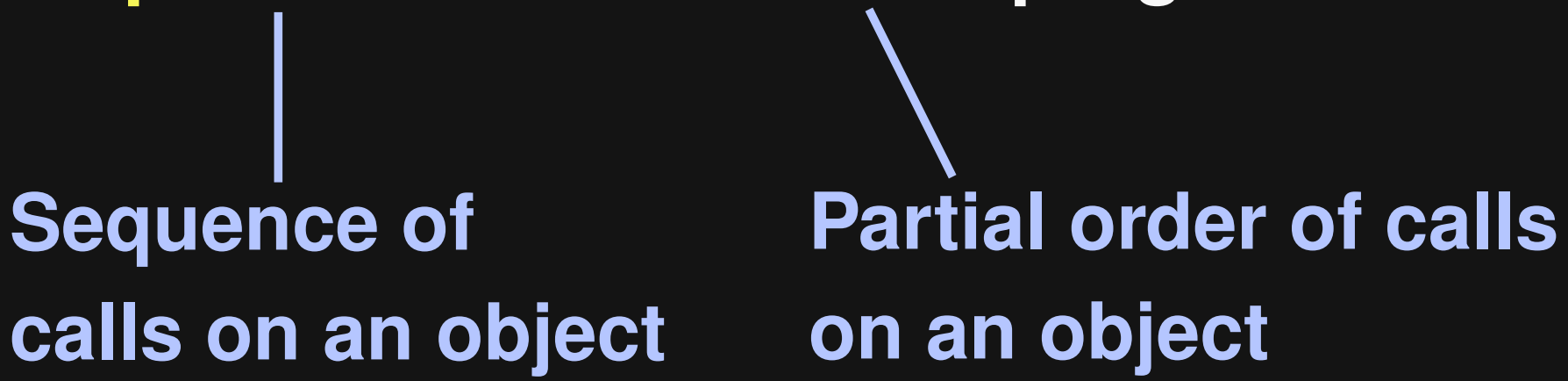
Substitutability

A subclass object should behave like a superclass object when being used through the superclass type.

[Liskov1987]

Sequential + Concurrent

Substitutability: Matters in sequential and concurrent programs



The Problem

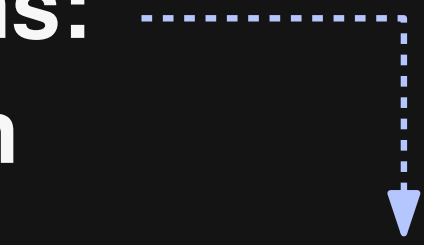
How to enforce substitutability?

- **Language restrictions:**
  Not powerful enough
- **Verification:**
  Not practical

The Problem

How to enforce substitutability?

- **Language restrictions:**
  Not powerful enough
- **Verification:**
  Not practical



```
class Super {
  m(Object o) {..}
}
class Sub extends Super {
  m(Foo o) {..}
}
```

The Problem

How to enforce substitutability?

- **Language restrictions:**
  Not powerful enough
- **Verification:**
  Not practical

The Problem

How to enforce substitutability?

- **Language restrictions:**
  Not powerful enough
- **Verification:**
  Not practical

In practice: 1/3 of all subclasses broken

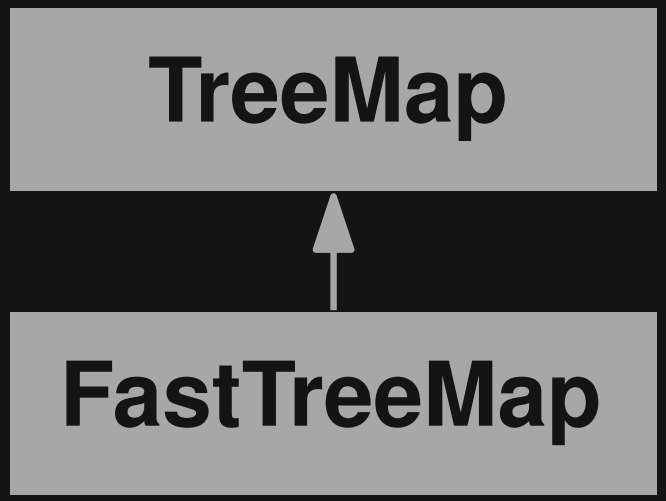
(for Java classes from 26 popular libraries)

Real-World Example

```
TreeMap m = ...
m.put(23, m);
m.pollLastEntry();
m.hashCode();
```

Real-World Example

```
TreeMap m = ...
m.put(23, m);
m.pollLastEntry();
m.hashCode();
```



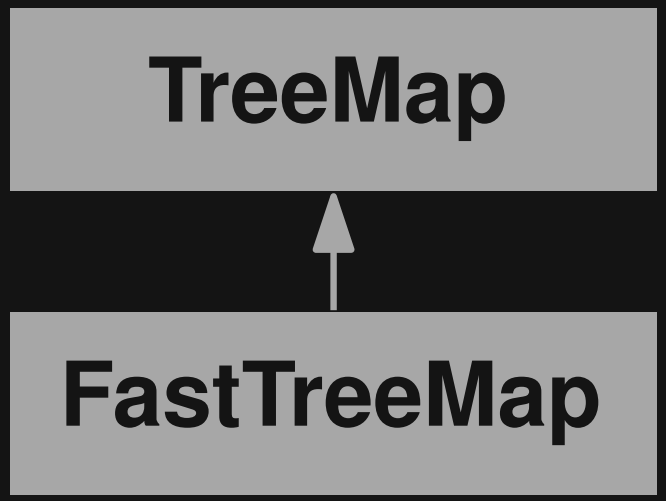
OK

StackOverflow-Error

Real-World Example

```
TreeMap m = ...
m.put(23, m);
m.pollLastEntry();
m.hashCode();
```

Problem:
May surprise clients of `TreeMap`



OK

`StackOverflow-Error`

This Talk

Automatic and precise

detection of unsafe substitutes

This Talk

Automatic and precise

detection of unsafe substitutes

Subclass that behaves differently from its superclass

This Talk

Only input:
Classes to test

Automatic and precise

detection of unsafe substitutes

Subclass that behaves
differently from its superclass

This Talk

Only input:
Classes to test

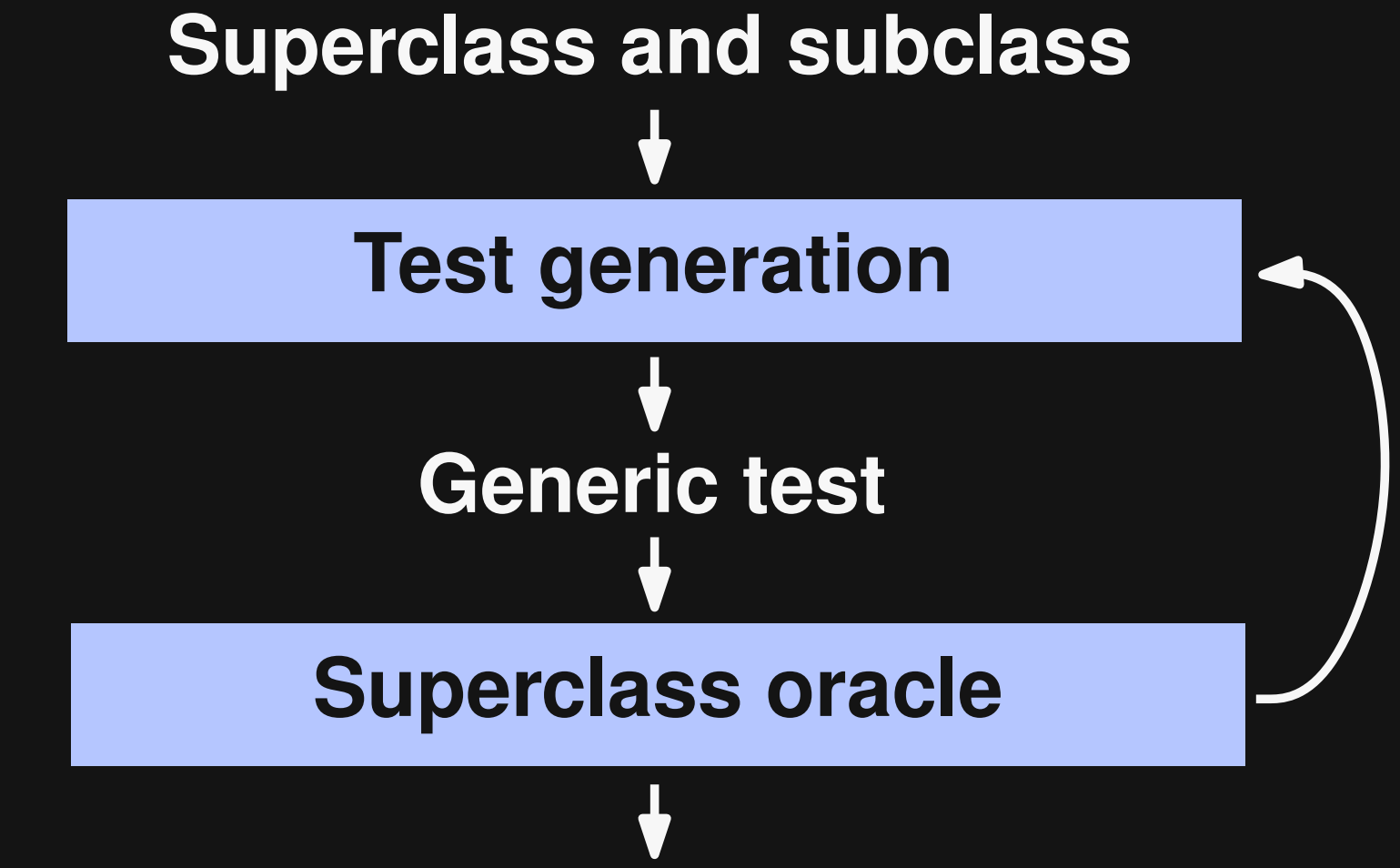
Only output:
Unsafe substitutes

Automatic and precise

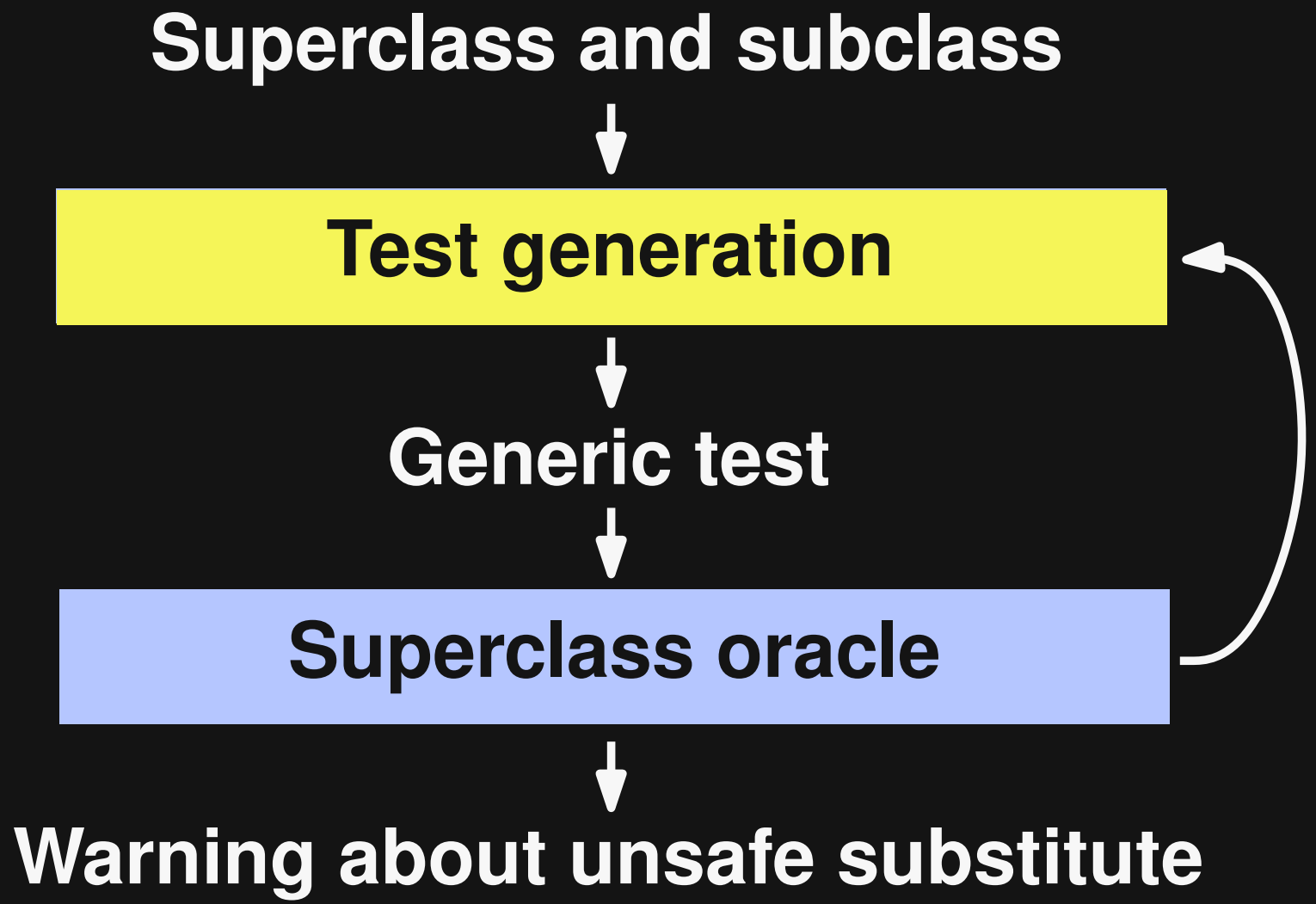
detection of unsafe substitutes

Subclass that behaves
differently from its superclass

Overview


Superclass and subclass
Test generation
Generic test
Superclass oracle
Warning about unsafe substitute

Overview


Superclass and subclass
Test generation
Generic test
Superclass oracle
Warning about unsafe substitute

Generic Tests

⋮ (sequence of calls to prepare arguments)

```
Super s = new Super(..) OR
          new Sub(..)
```

⋮ (calls to `s` from one or more threads)

Generic Tests

⋮ (sequence of calls to prepare arguments)

```
Super s = new Super(..) OR
          new Sub(..)
```

Run the same test with either class

⋮ (calls to `s` from one or more threads)

Generic Tests

⋮ (sequence of calls to prepare arguments)

```
Super s = new Super(..) OR
          new Sub(..)
```

⋮ (calls to `s` from one or more threads)

Generic Tests

⋮ (sequence of calls to prepare arguments)

```
Super s = new Super(..) OR
          new Sub(..)
```

⋮ (calls to `s` from one or more threads)

Test Generation

Feedback-directed, random generation of sequential and concurrent tests [PLDI 2012]

```
TreeMap m = new TreeMap() OR
            new FastTreeMap();
m.put(23, m);
m.pollLastEntry();
m.hashCode();
```

Randomly selected methods with random arguments

Challenge: Constructors

```
TreeMap m = new TreeMap(map1) OR
            new FastTreeMap(map2)
```

Goal: **Semantically equivalent** Super and Sub instances

Problem: **Constructors** are **not inherited** in Java

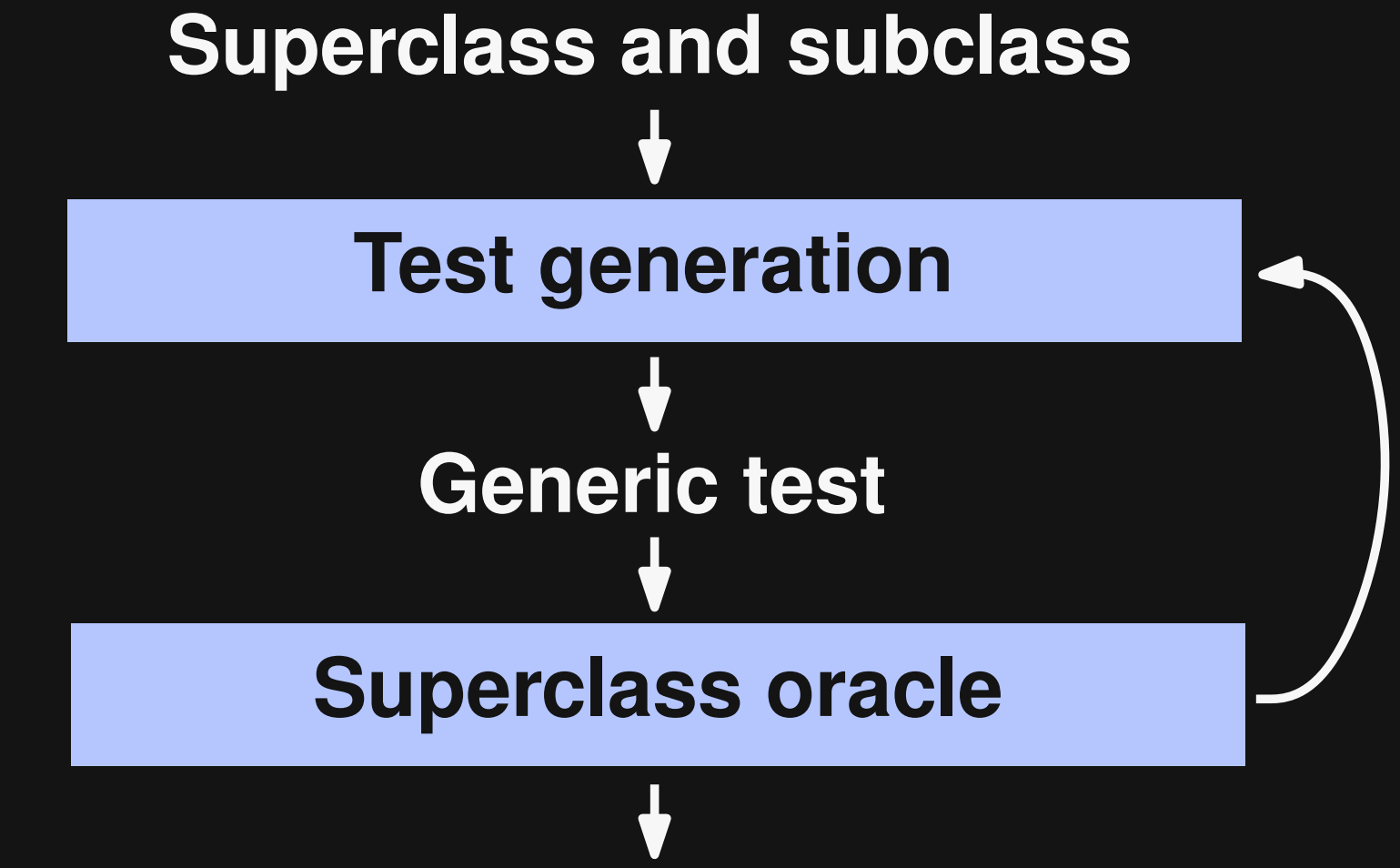
Constructor Mappings

Heuristic:

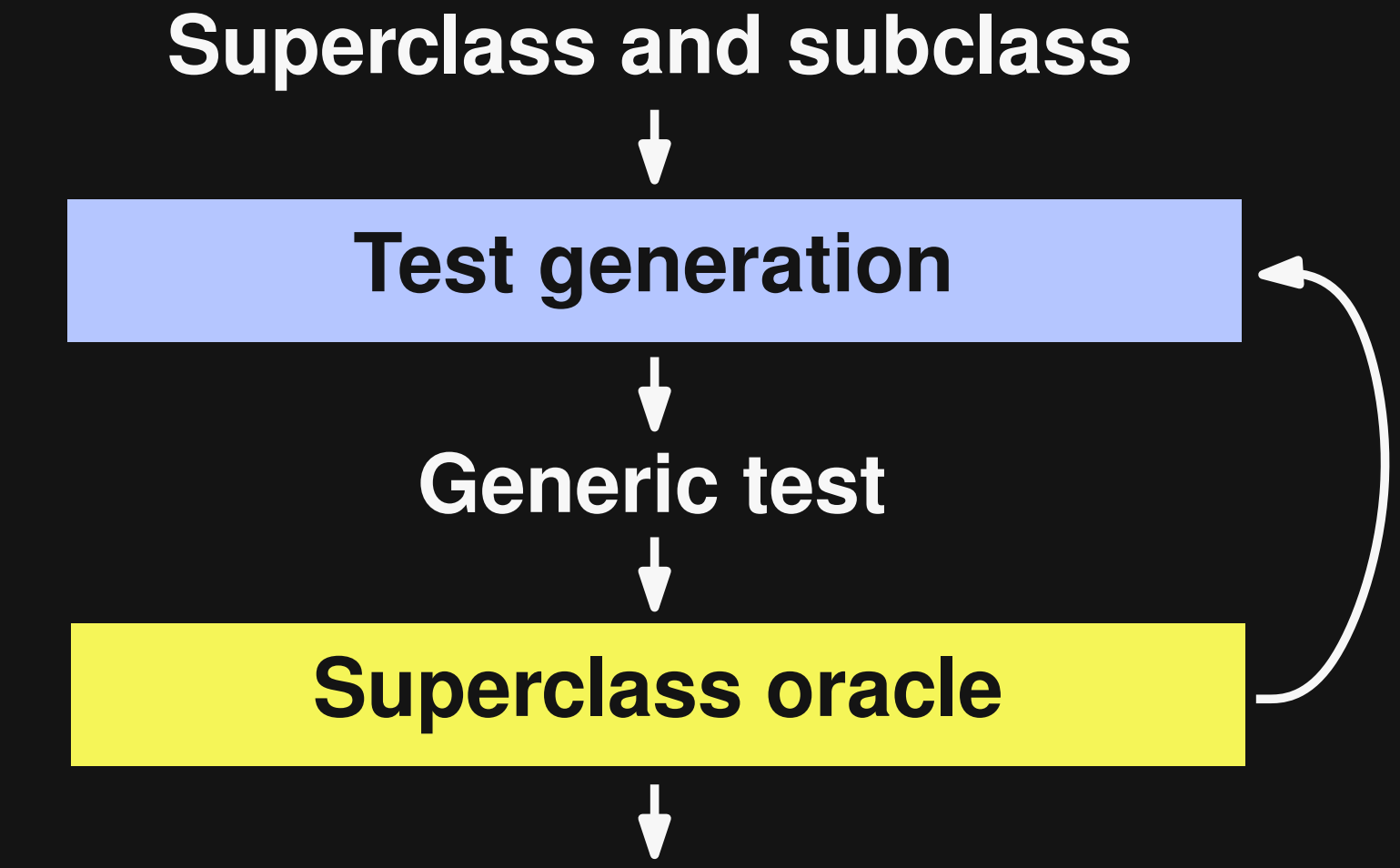
Map constructors by argument types and pass same arguments

- `Super(int, Foo)` ≡ `Sub(int, Foo)`
- `TreeMap()` ≡ `FastTreeMap()`

Overview


Superclass and subclass
Test generation
Generic test
Superclass oracle
Warning about unsafe substitute

Overview


Superclass and subclass
Test generation
Generic test
Superclass oracle
Warning about unsafe substitute

Superclass Oracle

Idea:

Warn if Sub's behavior diverges from Super's behavior

Sequential tests:
Compare two executions

Concurrent tests:
Compare two sets of executions

Output Oracle

Warn if return values differ

- Primitives & String: Compare values
- Reference values: Compare nullness

Output Oracle

Warn if return values differ

- Primitives & String: Compare values
- Reference values: Compare nullness

Example:

```
TreeMap m = new TreeMap();
m.put(23, m);       // null
m.pollLastEntry(); // non-null

TreeMap m = new FastTreeMap();
m.put(23, m);       // null
m.pollLastEntry(); // null
```

Output Oracle

Warn if return values differ

- Primitives & String: Compare values
- Reference values: Compare nullness

Example:

```
TreeMap m = new TreeMap();
m.put(23, m);          // null
m.pollLastEntry(); // non-null

TreeMap m = new FastTreeMap();
m.put(23, m);          // null
m.pollLastEntry(); // null
```

Warning

Crash Oracle

Warn if Sub leads to exception or deadlock, but Super doesn’t

Crash Oracle

Warn if Sub leads to exception or deadlock, but Super doesn't

Example:

```
TreeMap m = new TreeMap();
m.put(23, m);       // OK
m.pollLastEntry(); // OK
m.hashCode();       // OK

TreeMap m = new FastTreeMap();
m.put(23, m);       // OK
m.pollLastEntry(); // OK
m.hashCode();       // Exception
```

Crash Oracle

Warn if Sub leads to exception or deadlock, but Super doesn't

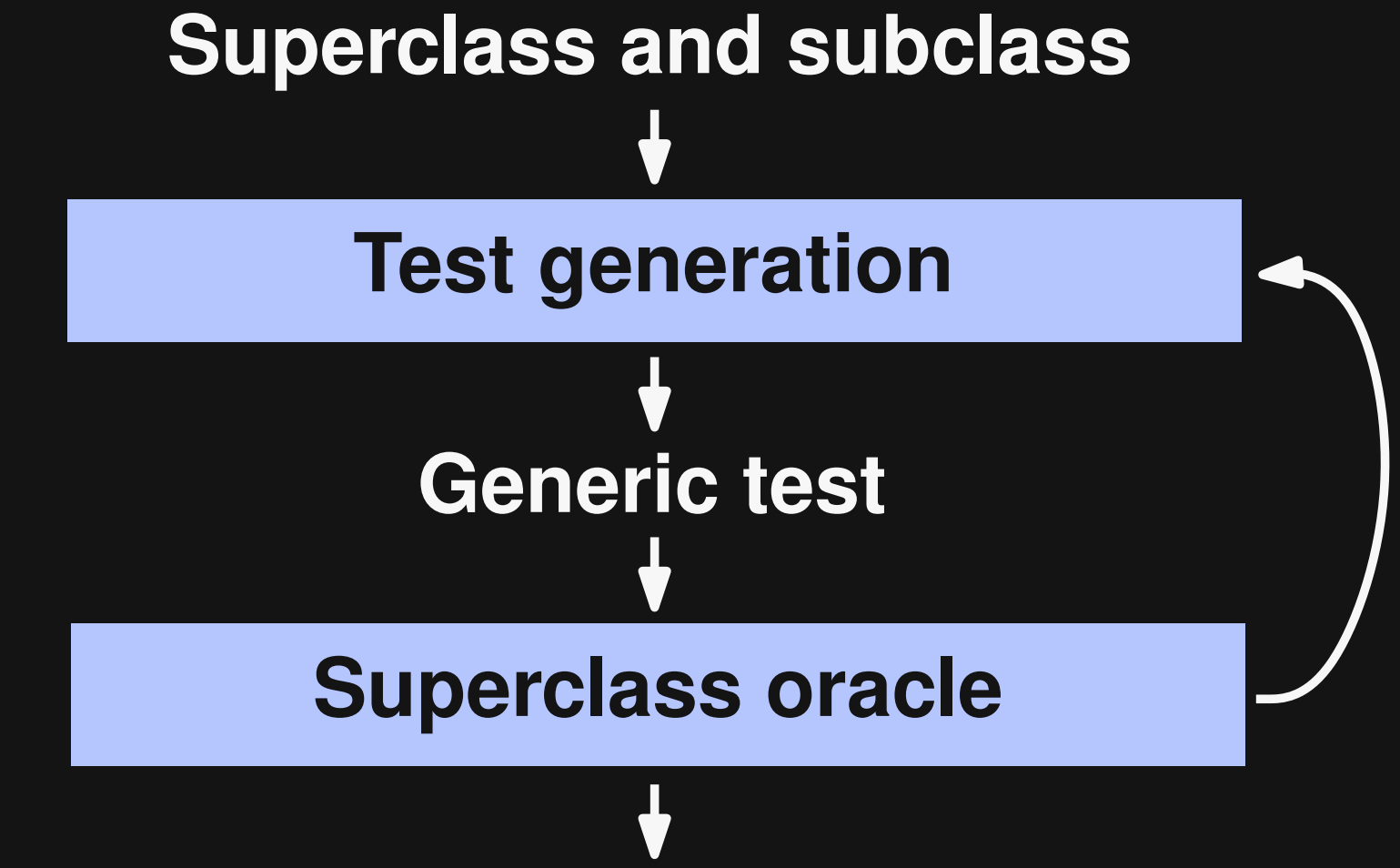
Example:

```
TreeMap m = new TreeMap();
m.put(23, m);          // OK
m.pollLastEntry();     // OK
m.hashCode();          // OK

TreeMap m = new FastTreeMap();
m.put(23, m);          // OK
m.pollLastEntry();     // OK
m.hashCode();          // Exception
```

Warning

Overview


Superclass and subclass
Test generation
Generic test
Superclass oracle
Warning about unsafe substitute

Evaluation: Setup

145 class pairs from
26 real-world Java libraries

- 116 sequentially used
- 29 concurrently used
- Apache Commons Collections, dom4j, iText, and libraries in Qualitas corpus

Stop testing after a fixed number of tests

Results: Output Oracle

42% of all subclasses are **output-diverging substitutes**

Most of them (93%) are **benign**

Example: Output-divergent

```
Namespace ns = new Namespace("a", "b");
boolean b = ns.supportsParent();  // false
```

```
Namespace ns = new DefaultNamespace("a", "b");
boolean b = ns.supportsParent();  // true
```

Example: Output-divergent

```
Namespace ns = new Namespace("a", "b");
boolean b = ns.supportsParent();  // false
```

```
Namespace ns = new DefaultNamespace("a", "b");
boolean b = ns.supportsParent();  // true
```

Different behavior but not a bug

Reasons for False Positives

- **Ad-hoc reflection**
- **String representations differ**
- **Constructor mismatch: Heuristic fails**

Results: Crash Oracle

30% of all subclasses are **crashing substitutes**

All issues are **bugs** that should be fixed

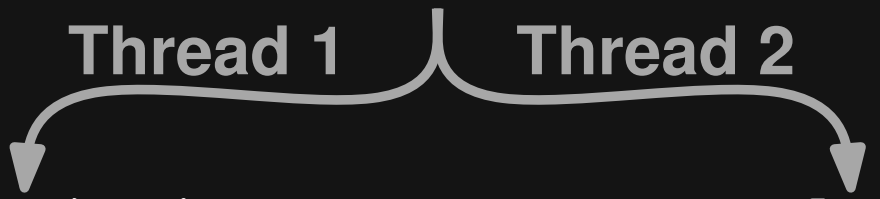
Example: Crashing (1)

```
TreeMap m = new TreeMap();
m.put(23, m);          // OK
m.pollLastEntry(); // OK
m.hashCode();          // OK

TreeMap m = new FastTreeMap();
m.put(23, m);          // OK
m.pollLastEntry(); // OK
m.hashCode();          // Exception
```

Example: Crashing (2)

```
Properties p = new Properties();
p.setProperty("a", "b");
```

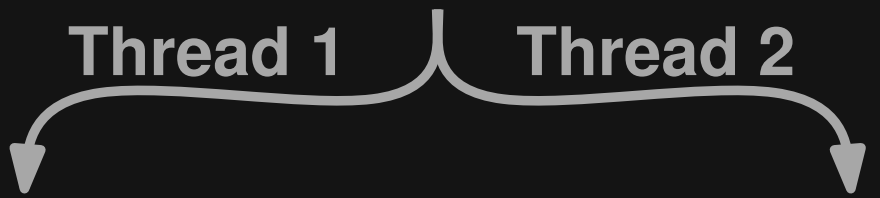



```
p.remove("a");
```

```
p.clear();
```

OK

```
Properties p = new PropertyMap();
p.setProperty("a", "b");
```




```
p.remove("a");
```

```
p.clear();
```

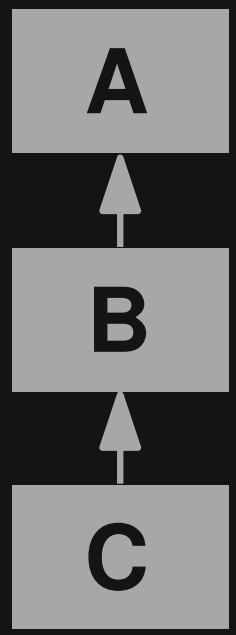
Exception

Root Causes for Bugs

- Sub imposes stronger precondition
- Sub removes methods
  (`UnsupportedOperationException`)
- Sub removes synchronization
- Propagated unsafety

Root Causes for Bugs

- Sub imposes stronger precondition
- Sub removes methods
  (`UnsupportedOperationException`)
- Sub removes synchronization
- Propagated unsafety



Feedback from Developers

- **Reported 10 bugs**
  (e.g., JBoss, Commons Collections)

  $\rightarrow$ **8 of them fixed by now**

- **3 other bugs found and fixed independently of us**

Conclusion

Substitutability: Broken in practice

Automatic testing approach

- Crash oracle: Only real bugs

Need better language support for avoiding substitutability problems

Automatic Testing of Sequential and Concurrent Substitutability

Thank you!

Michael Pradel

Artifacts for download:
http://mp.binaervarianz.de/icse2013/