Synthesis of Equivalent Method Calls in Guava

Andrea Mattavelli and Alberto Goffi

University of Lugano, Switzerland

Alessandra Gorla

IMDEA Software Institute, Spain

Equivalence in Software

Google Guava

```
MultiMap m = new MultiMap();

//…
//check if element is already in map
if (m.contains(x))



//…
//add key-value in the map
m.put(k, v)
```

Equivalence in Software

Google Guava

```
MultiMap m = new MultiMap();

//…
//check if element is already in map
if (m.contains(x))
if (m.elementSet().contains(x))
if (m.count(x) > 0)

//…
//add key-value in the map
m.put(k, v)
m.putAll(k, Arrays.asList(v))
```

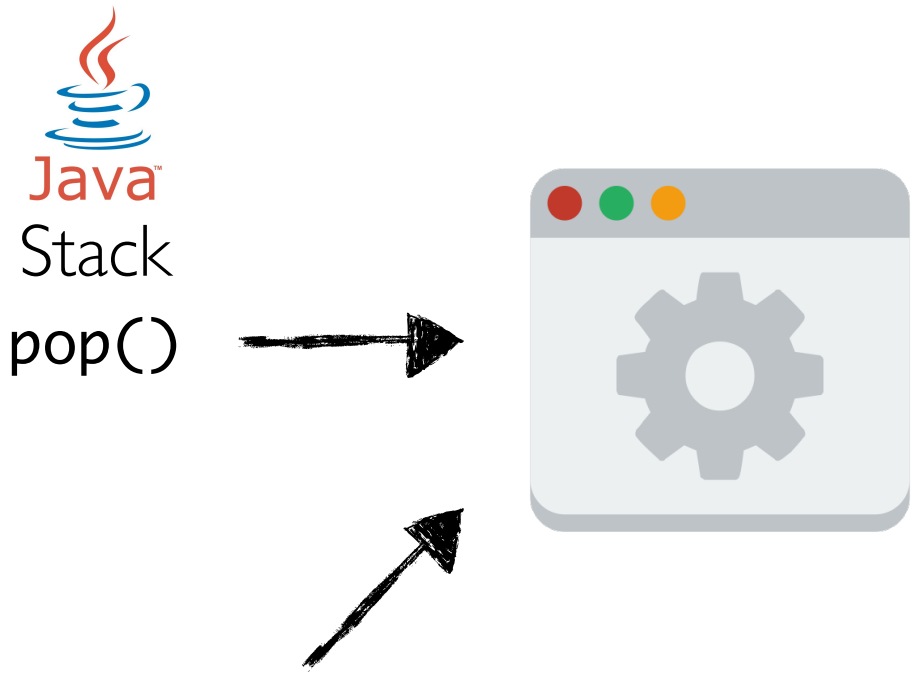
Exploiting Redundancy

Automatic repair

Test oracles

Security

Search-based Synthesis of Equivalences



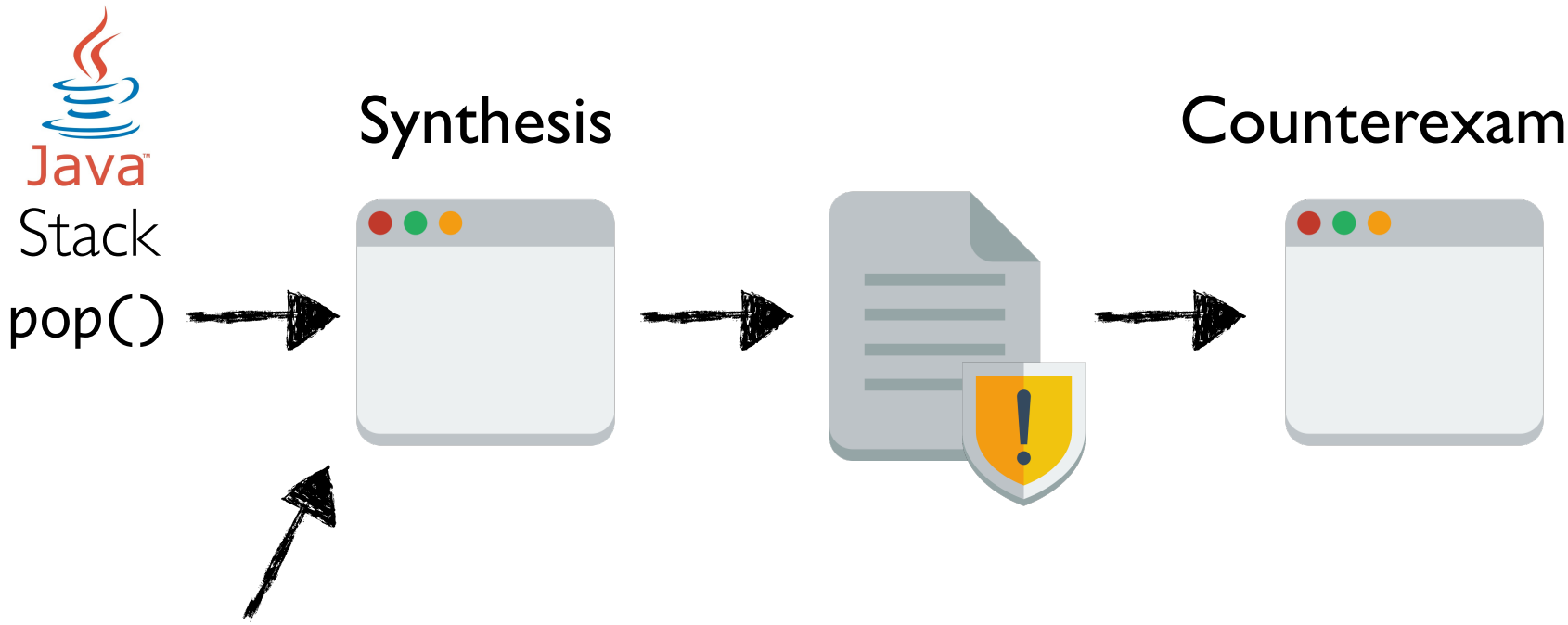

```
int el = s.peek();
int index = s.size();
index = index - 1;
s.remove(index);
return el;
```

Execution scenarios

```
Stack s = new Stack();
s.push(1);
s.push(1);
Object ret = s.pop();
```

```
Stack s = new Stack();
s.push(-4);
Object ret = s.pop();
```

Search-based Synthesis of Equivalences

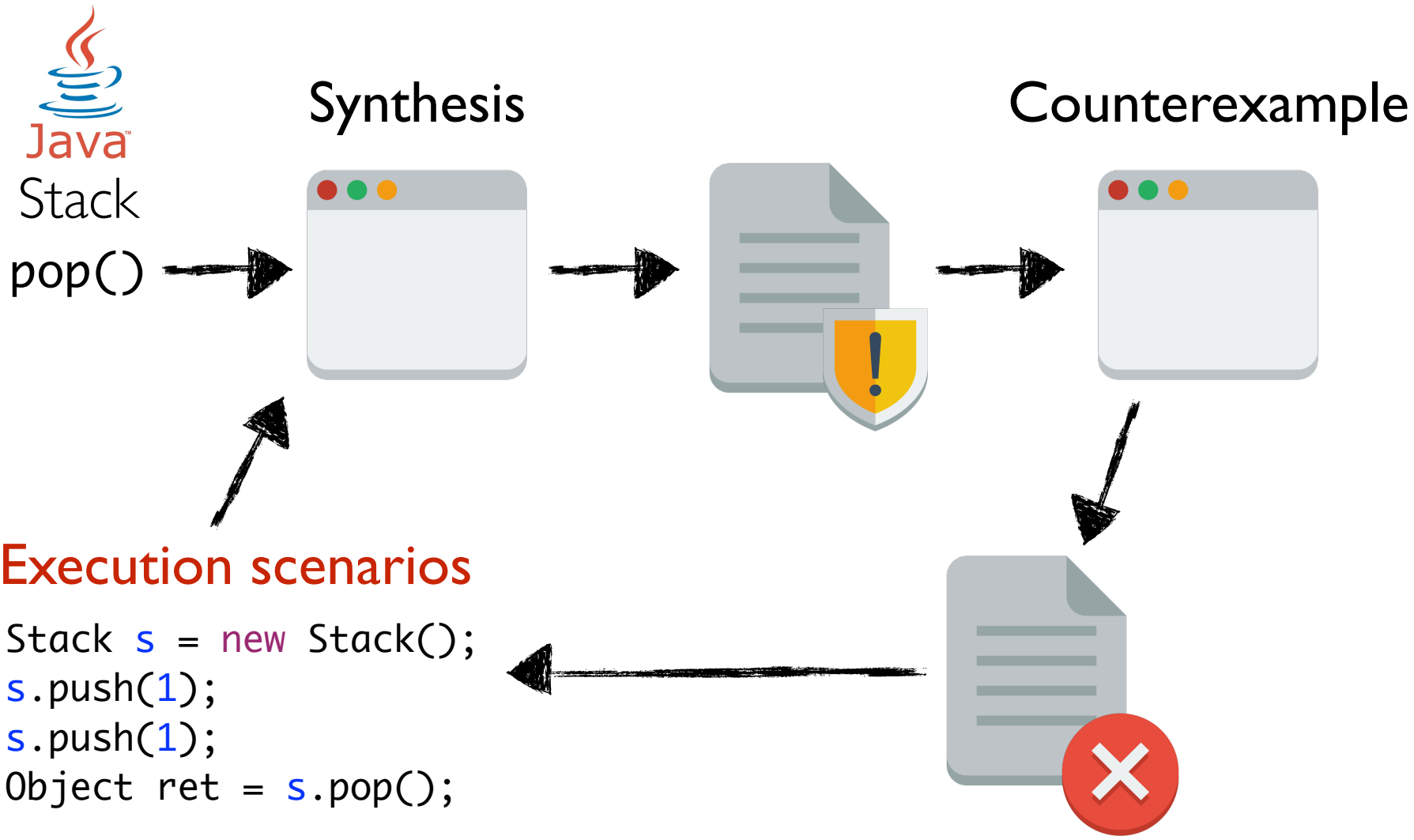


Execution scenarios

```
Stack s = new Stack();
s.push(1);
s.push(1);
Object ret = s.pop();
```

```
Stack s = new Stack();
s.push(-4);
Object ret = s.pop();
```

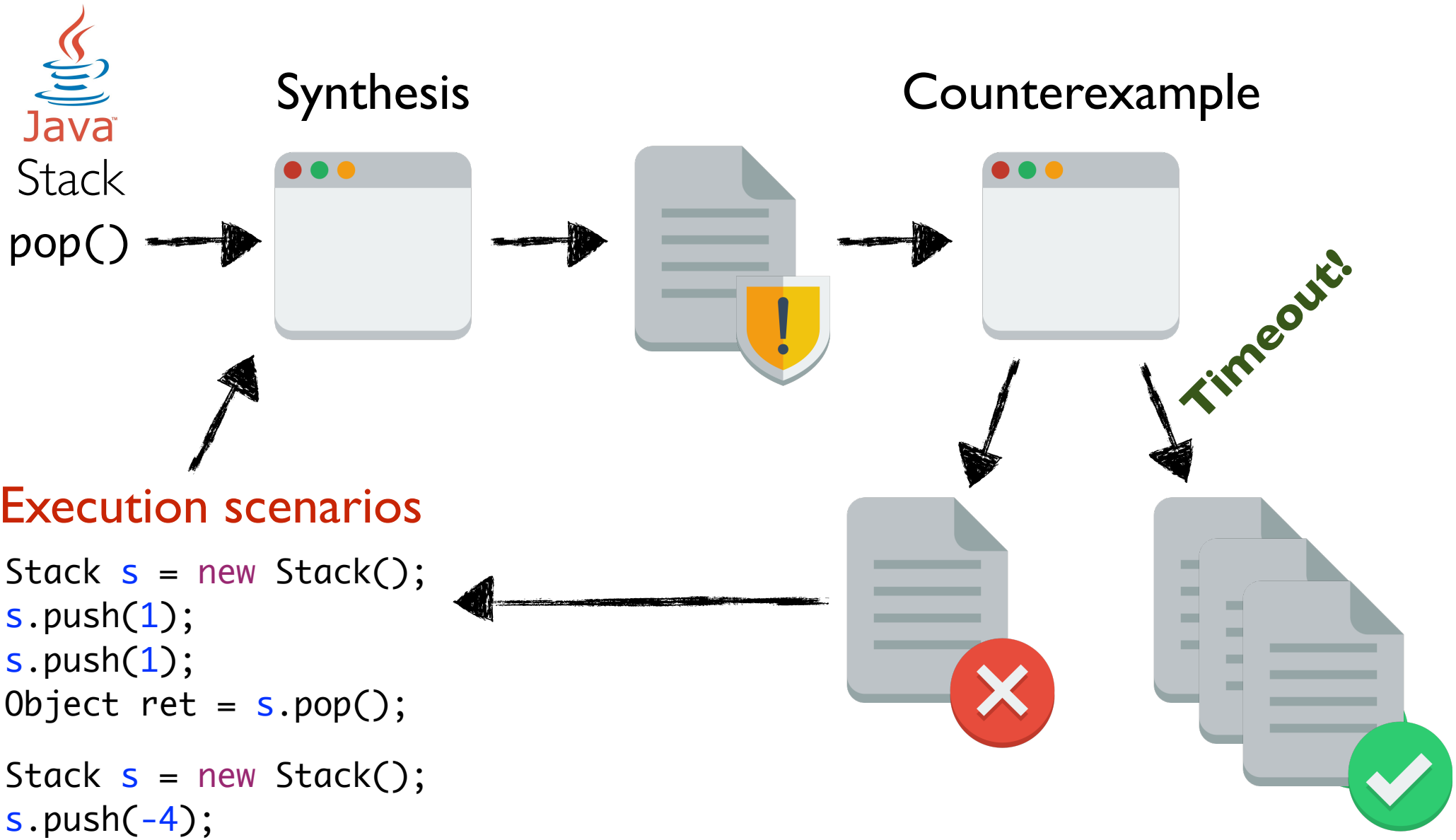
Search-based Synthesis of Equivalences



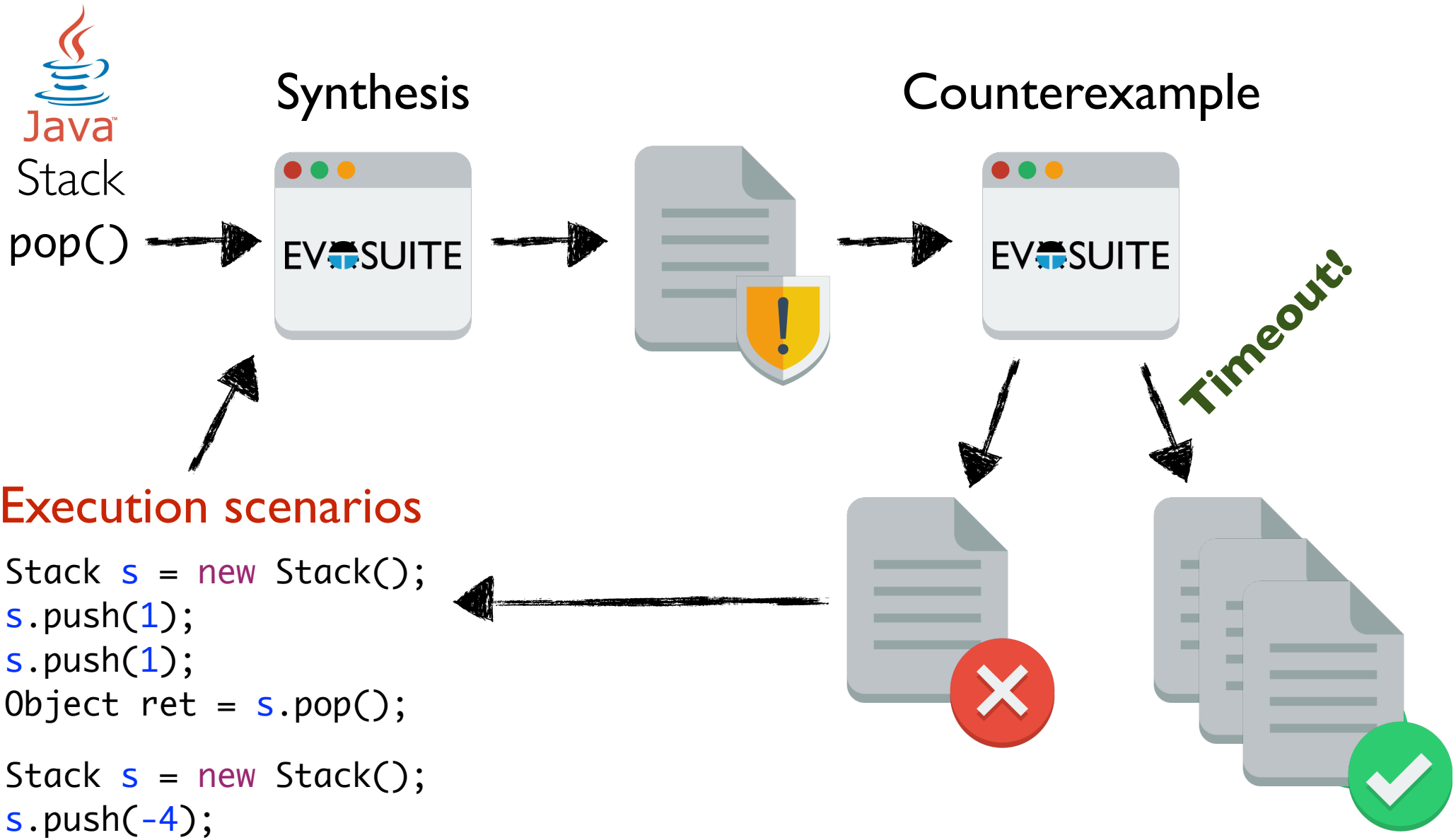

```
Stack s = new Stack();
s.push(1);
s.push(1);
Object ret = s.pop();
```

```
Stack s = new Stack();
s.push(-4);
Object ret = s.pop();
```

Search-based Synthesis of Equivalences



Search-based Synthesis of Equivalences



Search-Based Synthesis of Equivalent Method Sequences

Alberto Goffi[†], Alessandra Gorla[‡], Andrea Mattavelli[†], Mauro Pezzè[†*], and Paolo Tonella[§]

[†]University of Lugano, Switzerland, {goffia, mattavea, pezzem}@usi.ch
[‡]Saarland University, Germany, gorla@st.cs.uni-saarland.de
[§]Fondazione Bruno Kessler, Italy, tonella@fbk.eu

ABSTRACT

Software components are usually redundant, since their interface offers different operations that are *equivalent* in their functional behavior. Several reliability techniques exploit this redundancy to either detect or tolerate faults in software. Metamorphic testing, for instance, executes pairs of sequences of operations that are exp... alent results, and identifies faults ... outcomes. Some popular fault tol... techniques execute redundant oper... avoid failures at runtime. The comm... techniques, though, is that such redu... This means that the set of operatio... be equivalent in a given component should be available in the specifications. Unfortunately, inferring this information manually can be expensive and error prone.

This paper proposes a search-based technique to synthesize sequences of method invocations that are equivalent to a target method within a finite set of execution scenarios. The experimental results obtained on 47 methods from 7 classes show that the proposed approach correctly identifies equivalent method sequences in the majority of the cases where redundancy was known to exist, with very few false positives.

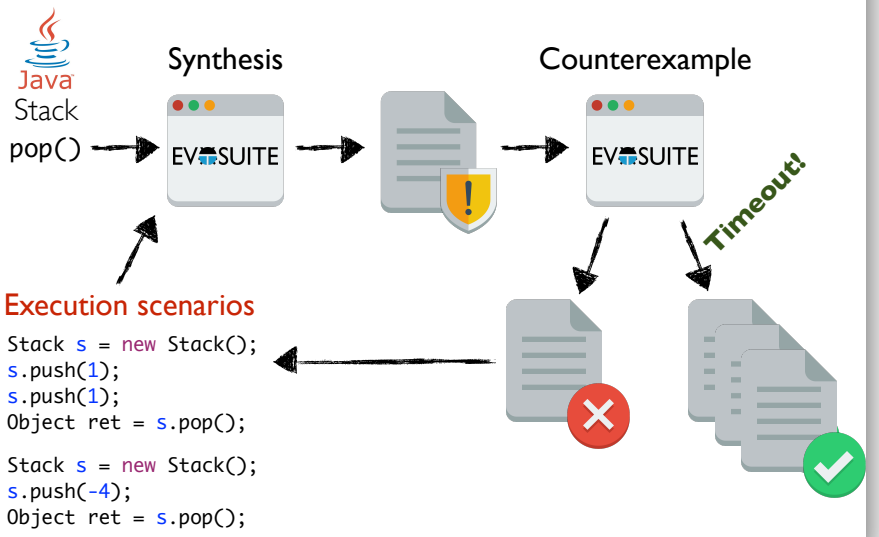
FSE 2014

1. INTRODUCTION

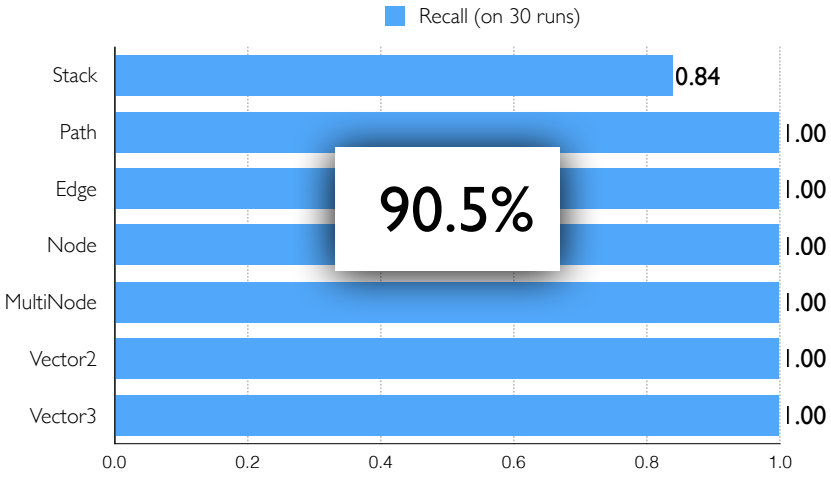
The presence of *equivalent* code fragments, for example methods or method sequences, make modern software systems *redundant*. Informally, two methods are equivalent if they produce indistinguishable results when called with proper parameters. This is the case, for instance, of methods ... Google Guava class AbstractMul... distinguishable results when putAll() ... containing the single value passed ...geable methods, as in the previous ...o obtain equivalent executions by ...l invocations. For example, method ... Java standard library is equivalent to the method sequence remove(size()-1). Indeed, removing the element on top of the stack (pop()) leads to the same result as removing the element in the last position (remove(size()-1)).

This form of redundancy should not be confused with what are usually referred to as code clones. Code clones are typically the result of bad design and implementation practices, such as copy and paste, and indicate the need of code refactoring [16]. Instead, the redundancy described above is the result of good design practice, as it aims to offer a richer API to client components and to increase code reusability.

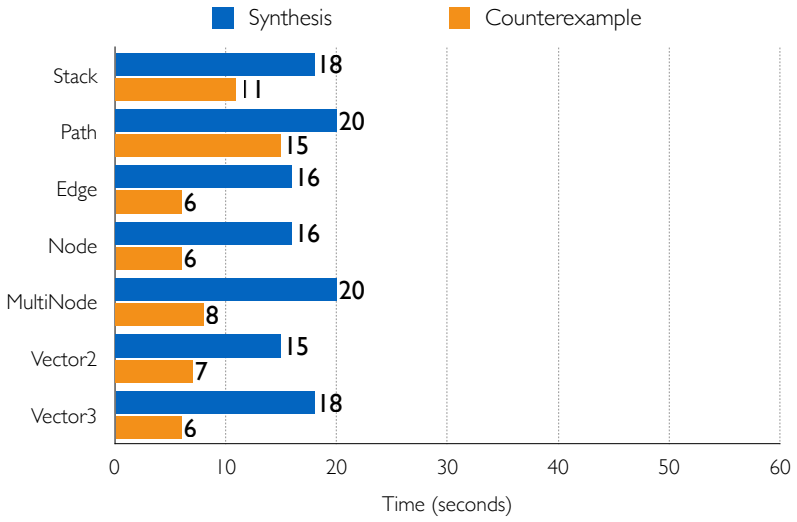
Search-based Synthesis of Equivalences



Effectiveness of Search-based Synthesis



Efficiency of Search-based Synthesis



Search-Based Synthesis of Equivalent Method Sequences

Alberto Goffi[†], Alessandra Gorla[‡], Andrea Mattavelli[†], Mauro Pezzè[†*], and Paolo Tonella[§]

[†]University of Lugano, Switzerland, {goffia, mattavea, pezzem}@usi.ch
[‡]Saarland University, Germany, gorla@st.cs.uni-saarland.de
[§]Fondazione Bruno Kessler, Italy, tonella@fbk.eu

ABSTRACT

Software components are usually redundant, since their interface offers different operations that are *equivalent* in their functional behavior. Several reliability techniques exploit this redundancy to either detect or tolerate faults in software. Metamorphic testing, for instance, executes pairs of sequences of operations that are exp... alent results, and identifies faults ... outcomes. Some popular fault tol... techniques execute redundant oper... avoid failures at runtime. The comm... techniques, though, is that such redu... This means that the set of operatio... be equivalent in a given component should be available in the specifications. Unfortunately, inferring this information manually can be expensive and error prone.

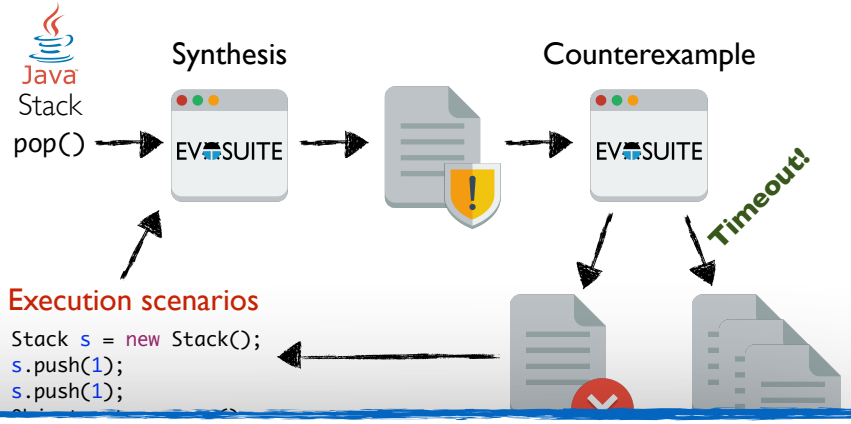
This paper proposes a search-based technique to synthesize

1. INTRODUCTION

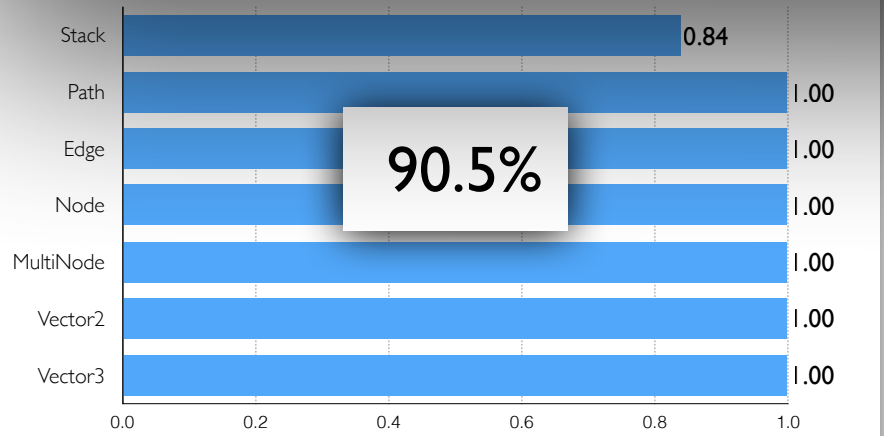
The presence of *equivalent* code fragments, for example methods or method sequences, make modern software systems *redundant*. Informally, two methods are equivalent if they produce indistinguishable results when called with proper parameters. This is the case, for instance, of methods ... Google Guava class AbstractMul... distinguishable results when putAll() ... containing the single value passed ...geable methods, as in the previous ...o obtain equivalent executions by ...l invocations. For example, method ... Java standard library is equivalent to the method sequence remove(size()-1). Indeed, removing the element on top of the stack (pop()) leads to the same result as removing the element in the last position (remove(size()-1)).

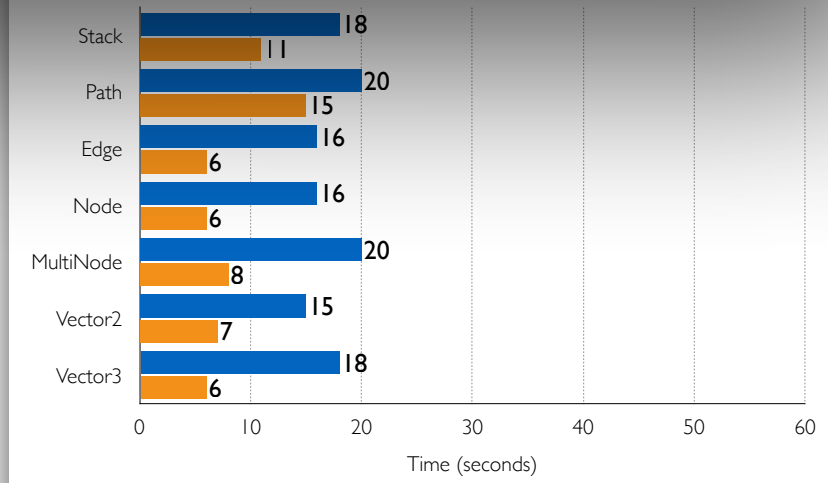
FSE 2014

Search-based Synthesis of Equivalences



Can SBES identify equivalences in Google Guava?





Google Guava

guava-libraries

Basic utilities

Collections

Caches

Functional idioms

Concurrency

Strings

Primitives

Ranges

I/O

Hashing

Math

Google Guava

guava-libraries

Basic utilities

Collections

Caches

Functional idioms

Concurrency

Strings

Primitives

Ranges

I/O

Hashing

Math

Google Guava: Challenges

Large Search Space

Generics Support

Google Guava: Large Search Space

guava-libraries

335 classes

5,400 methods

Google Guava: Large Search Space

```
addFirst(x)  ≡  addElementAt(x,0)
      pop()  ≡  remove(size()-1)
        x()  ≡  get(0)
        y()  ≡  get(1)
        z()  ≡  get(2)
```

Google Guava: Large Search Space

```
addFirst(x)  ≡  addElementAt(x,0)
      pop()  ≡  remove(size()-1)
        x()  ≡  get(0)
        y()  ≡  get(1)
        z()  ≡  get(2)
```

Memetic algorithms

Google Guava: Generics Support

```
class ArrayList<T>

class ContiguousSet<C extends Comparable>

class Condition<? super E>

class IterableToCollection<
  E,
  T extends Iterable<? extends E>,
  C extends Condition<? super E>,
  R extends Collection<E>
>
```

Google Guava: Generics Support

`Multimap<K, V>`

Execution scenarios

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(1, "String");
```

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(-4, "Test");
m.put(-4, "Test2");
```

Google Guava: Generics Support

```
Multimap<K, V>
```

Execution scenarios

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(1, "String");
```

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(-4, "Test");
m.put(-4, "Test2");
```

Google Guava: Generics Support

```
Multimap<K, V>
```

Execution scenarios

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(1, "String");
```

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(-4, "Test");
m.put(-4, "Test2");
```

```
put(K key, V value)
```

```
put(Integer key, String value)
```

Google Guava: Generics Support

```
Multimap<K, V>
```

Execution scenarios

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(1, "String");
```

```
Multimap<Integer, String> m = new Multimap<Integer, String>();
m.put(-4, "Test");
m.put(-4, "Test2");
```

```
put(K key, V value)
```

```
put(Integer key, String value)
```

Generics to concrete

Evaluation

Evaluation

guava-libraries

16 classes

220 methods

Evaluation

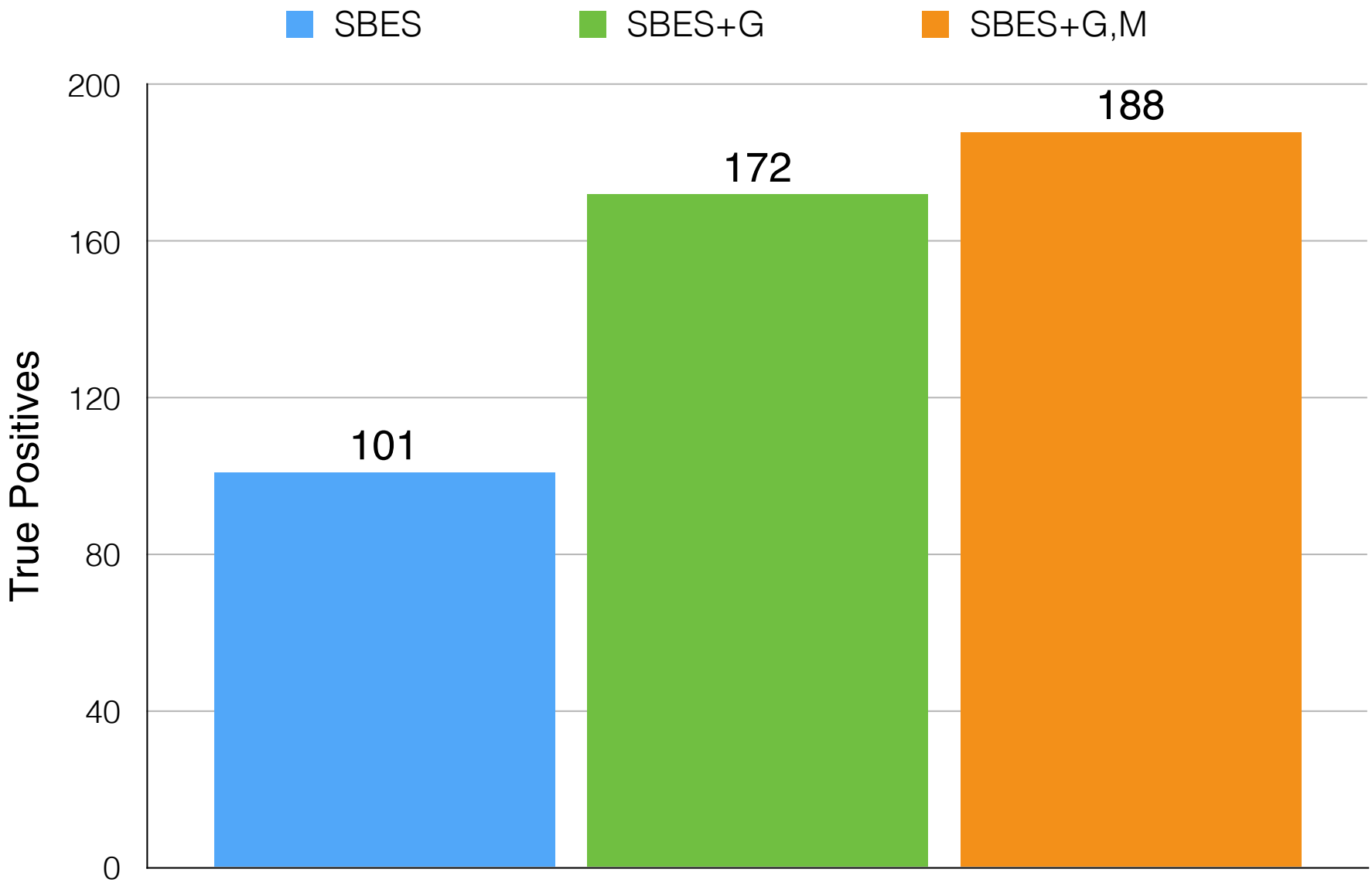
guava-libraries

16 classes

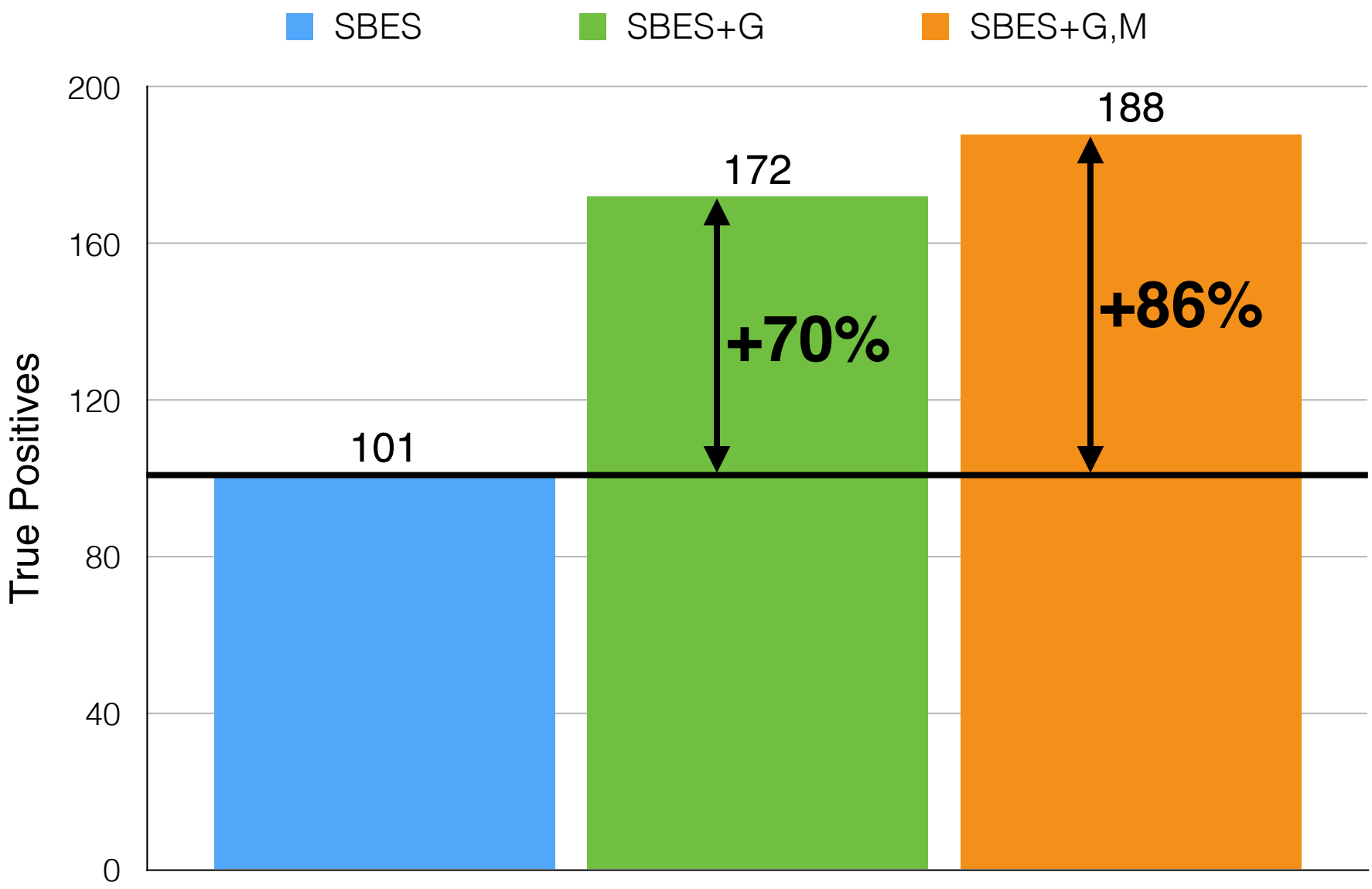
220 methods

SBES **SBES+G** **SBES+G,M**

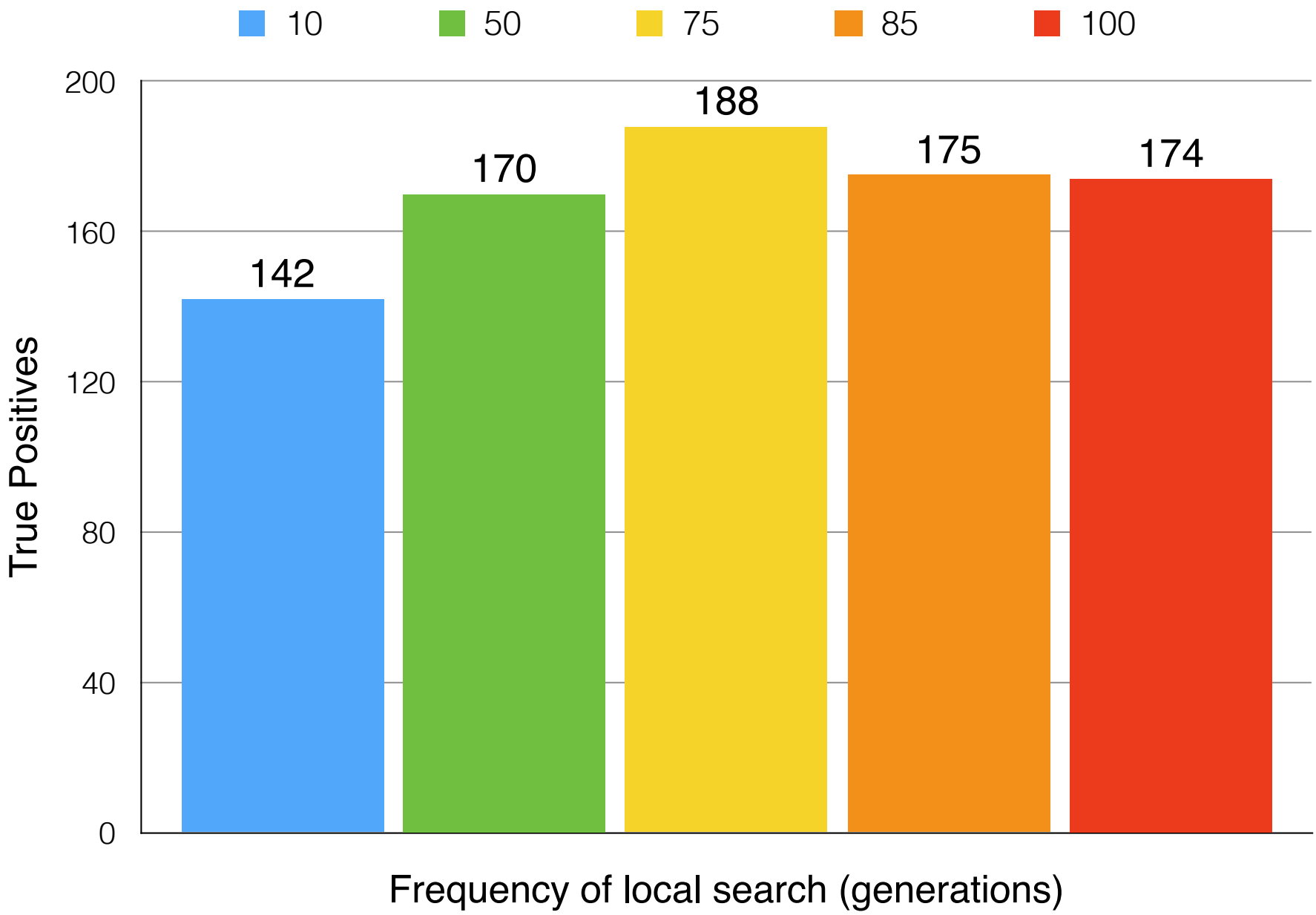
Effectiveness of Synthesis



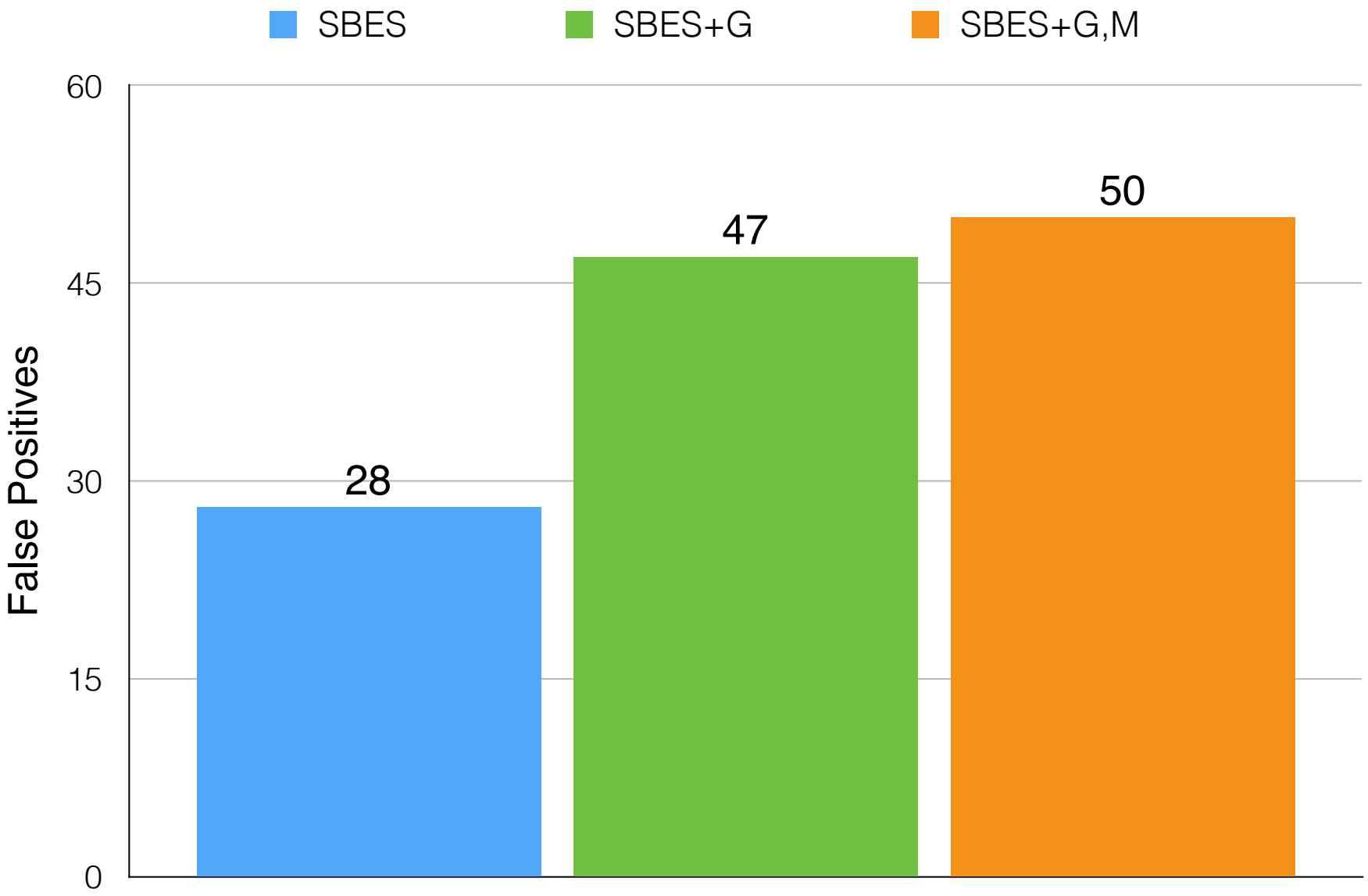
Effectiveness of Synthesis



Effectiveness of Memetic Algorithms



Effectiveness of Counterexamples



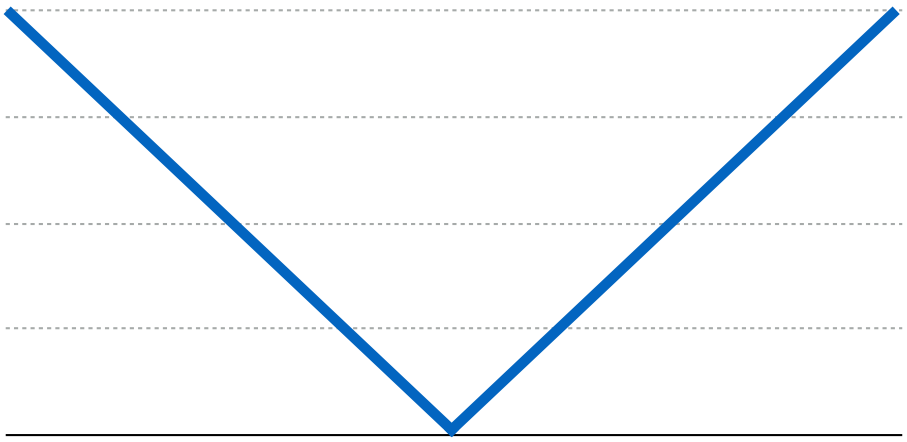
Effectiveness of Counterexamples

Synthesis

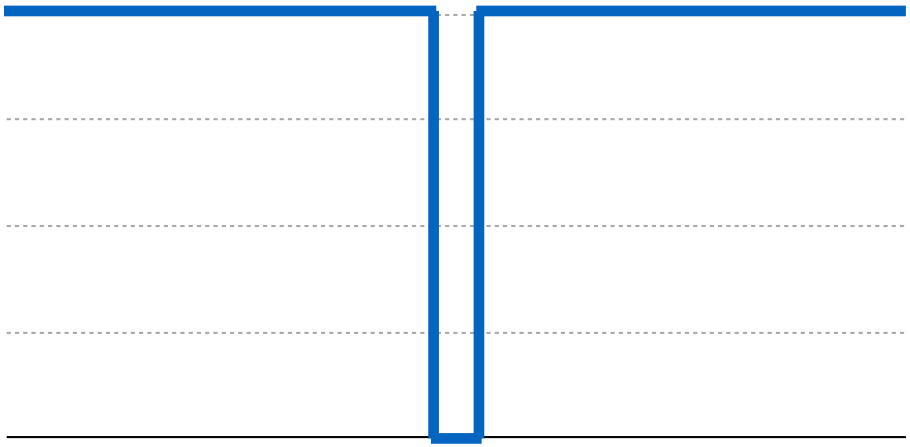
```
public void method_under_test() {
    if(distance(this,clone)==0 &&
         distance(expected,actual)==0) {
          // equivalent!
    }
}
```

Counterexample

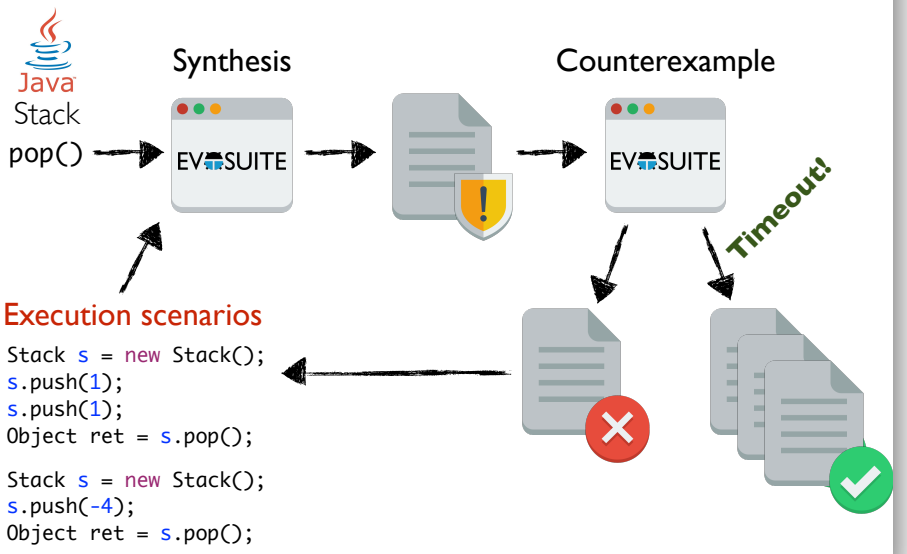
```
public void method_under_test() {
    if(distance(this,clone)>0 ||
         distance(expected,actual)>0) {
          // counterexample!
    }
}
```



Fitness value



Search-based Synthesis of Equivalences



Google Guava: Challenges

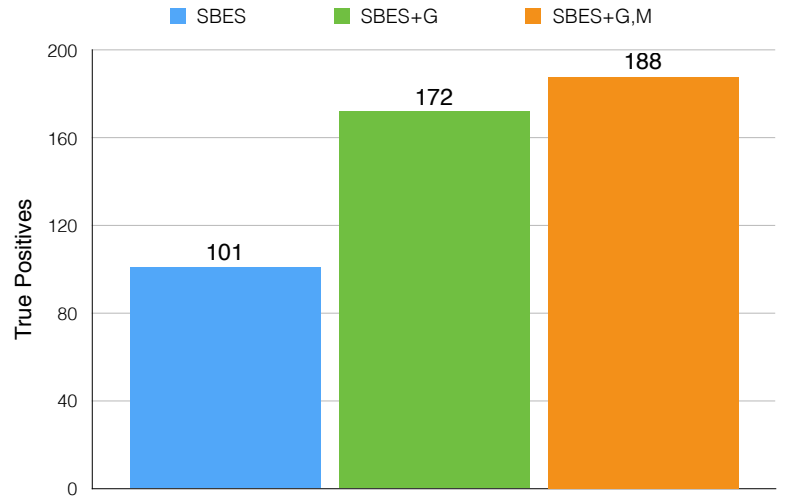
Large Search Space

Memetic algorithms

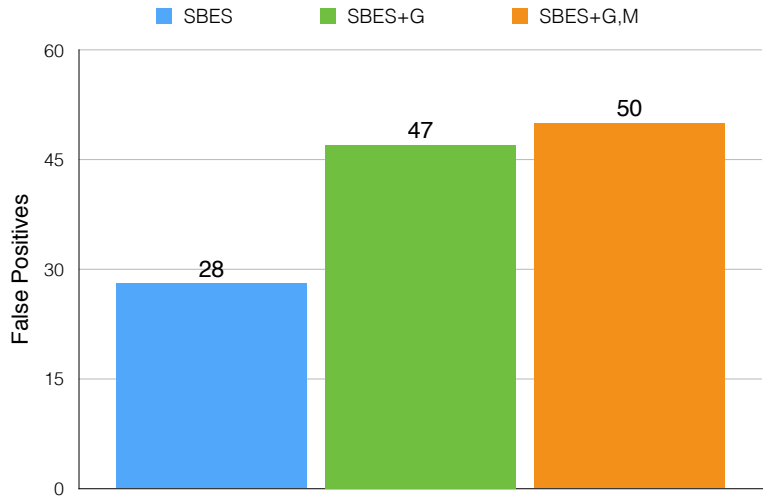
Generics Support

Generic-to-concrete

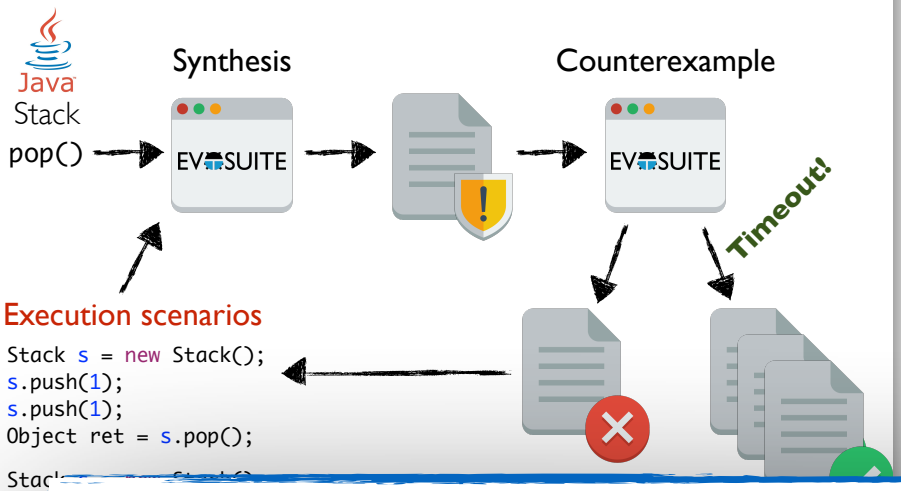
Effectiveness of Synthesis



Effectiveness of Counterexamples



Search-based Synthesis of Equivalences



Google Guava: Challenges

Large Search Space

Memetic algorithms

Generics Support

Generic-to-concrete

star.inf.usi.ch/sbes-challenge