Università della Svizzera italiana

Faculty of Informatics

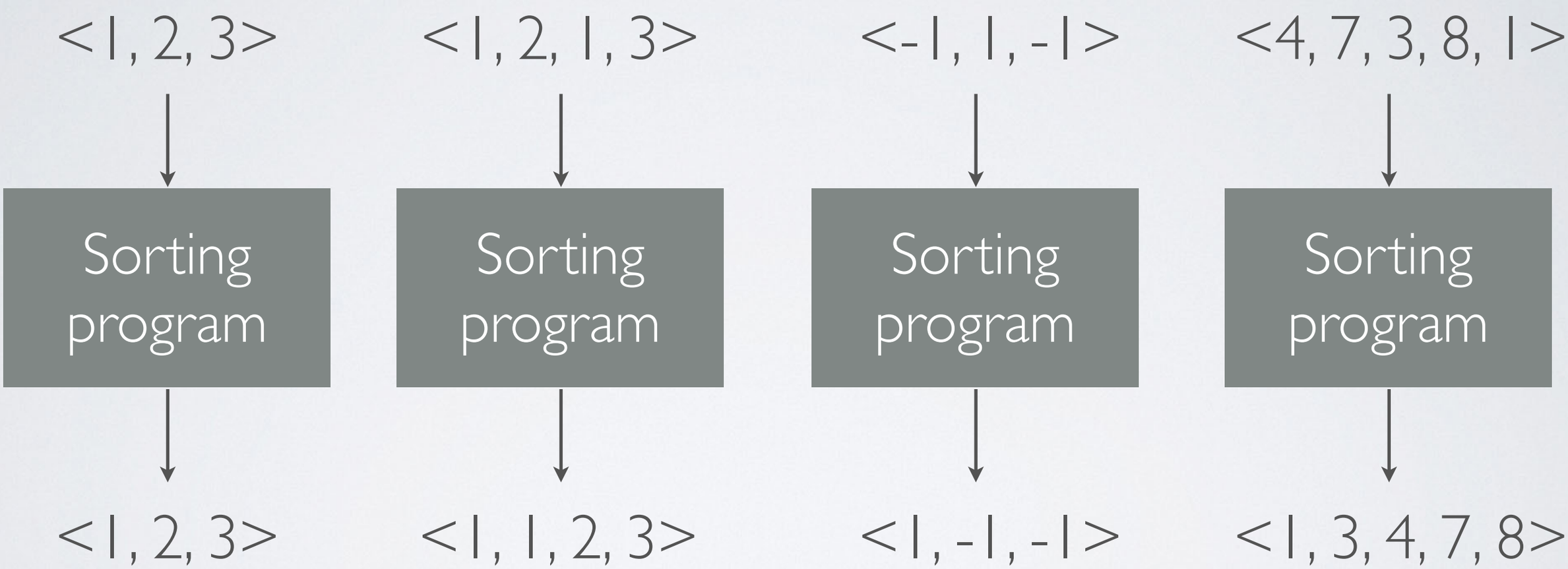
AUTOMATIC GENERATION OF COST-EFFECTIVE TEST ORACLES

Alberto Goffi
University of Lugano - Switzerland

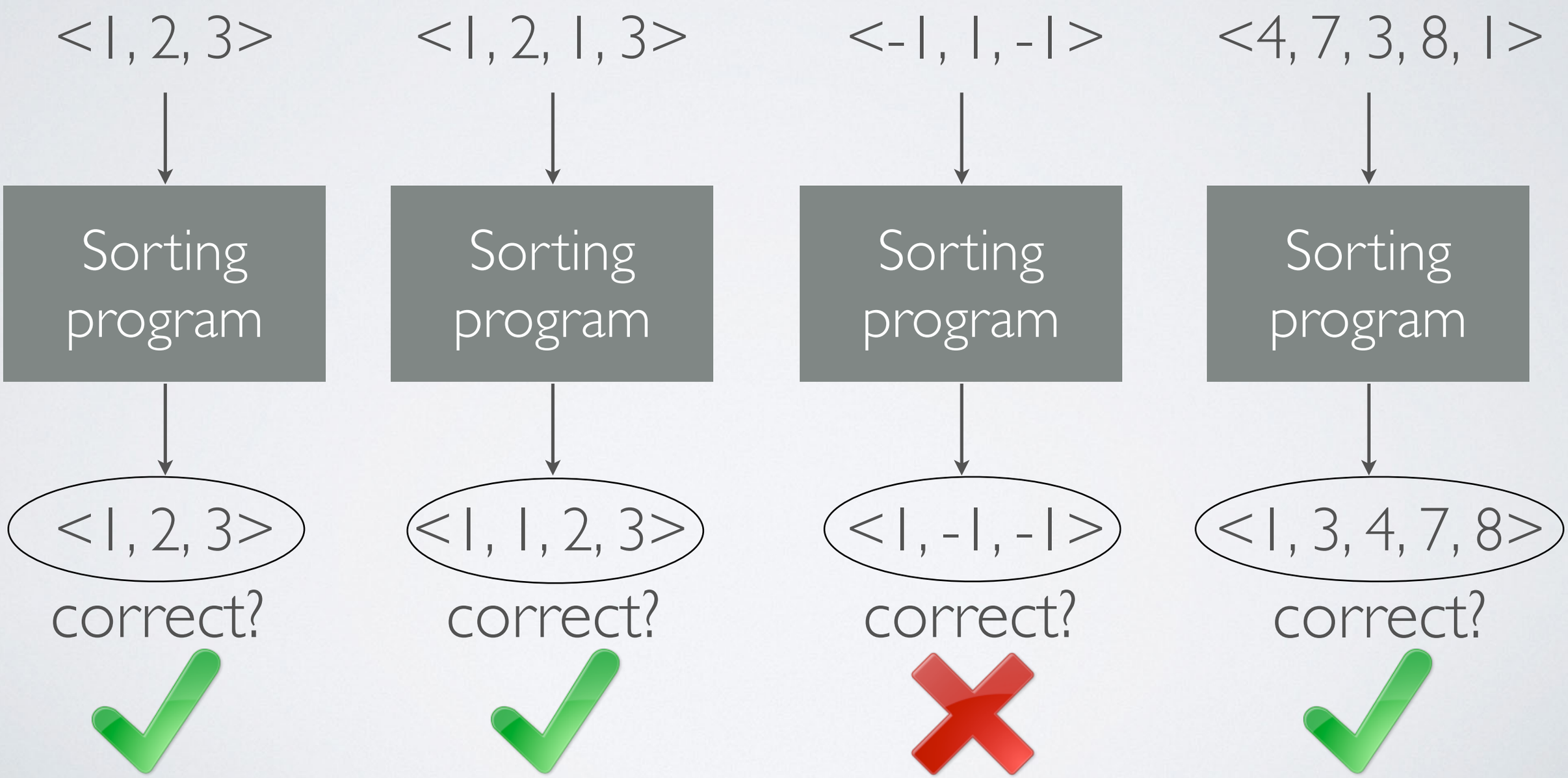
ICSE 2014 - Doctoral Symposium

PhD Advisor: Prof. Mauro Pezzè

EXAMPLE: TESTING A SORTING PROGRAM



EXAMPLE: TESTING A SORTING PROGRAM



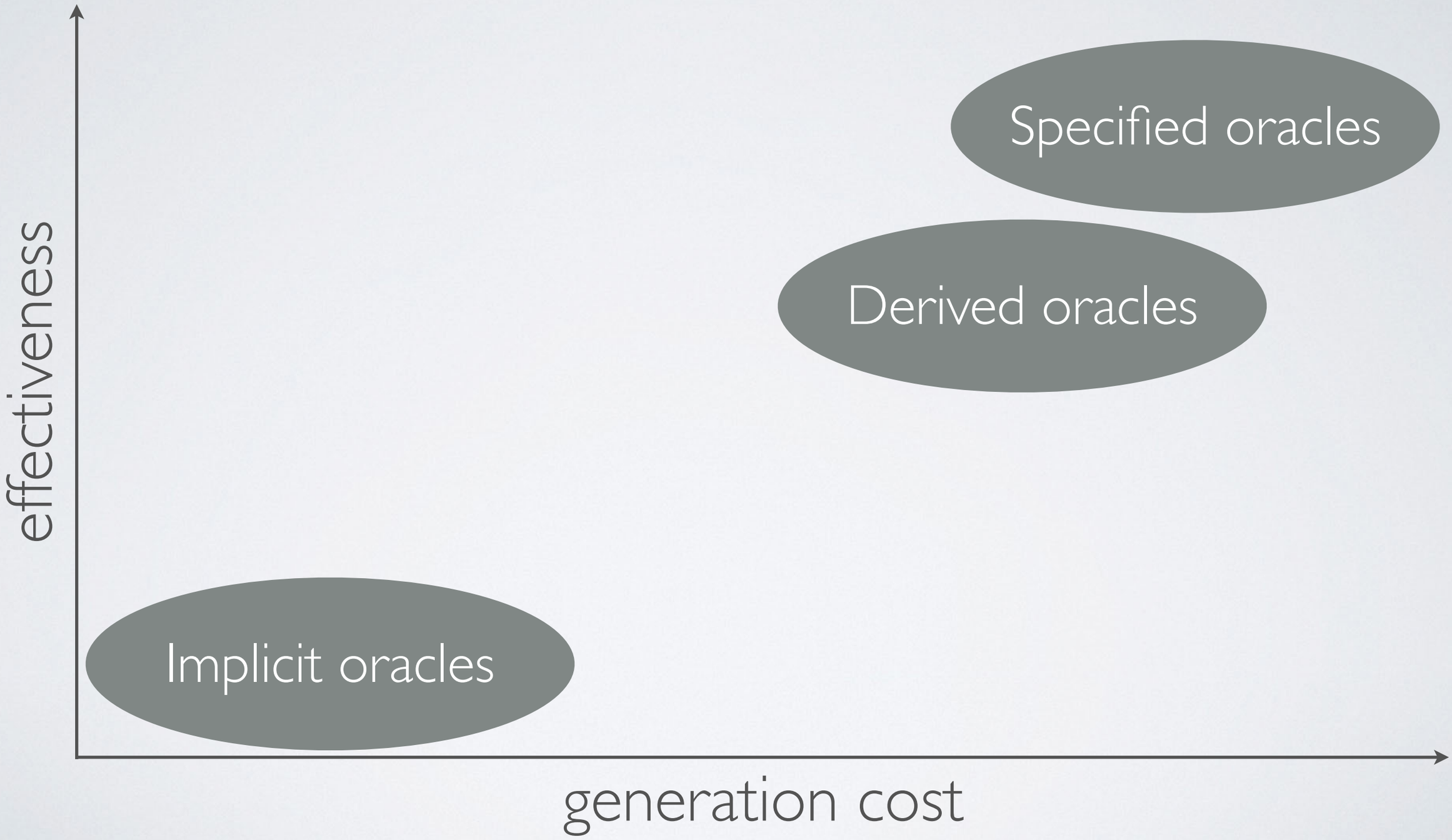
TEST ORACLES

Specified oracles

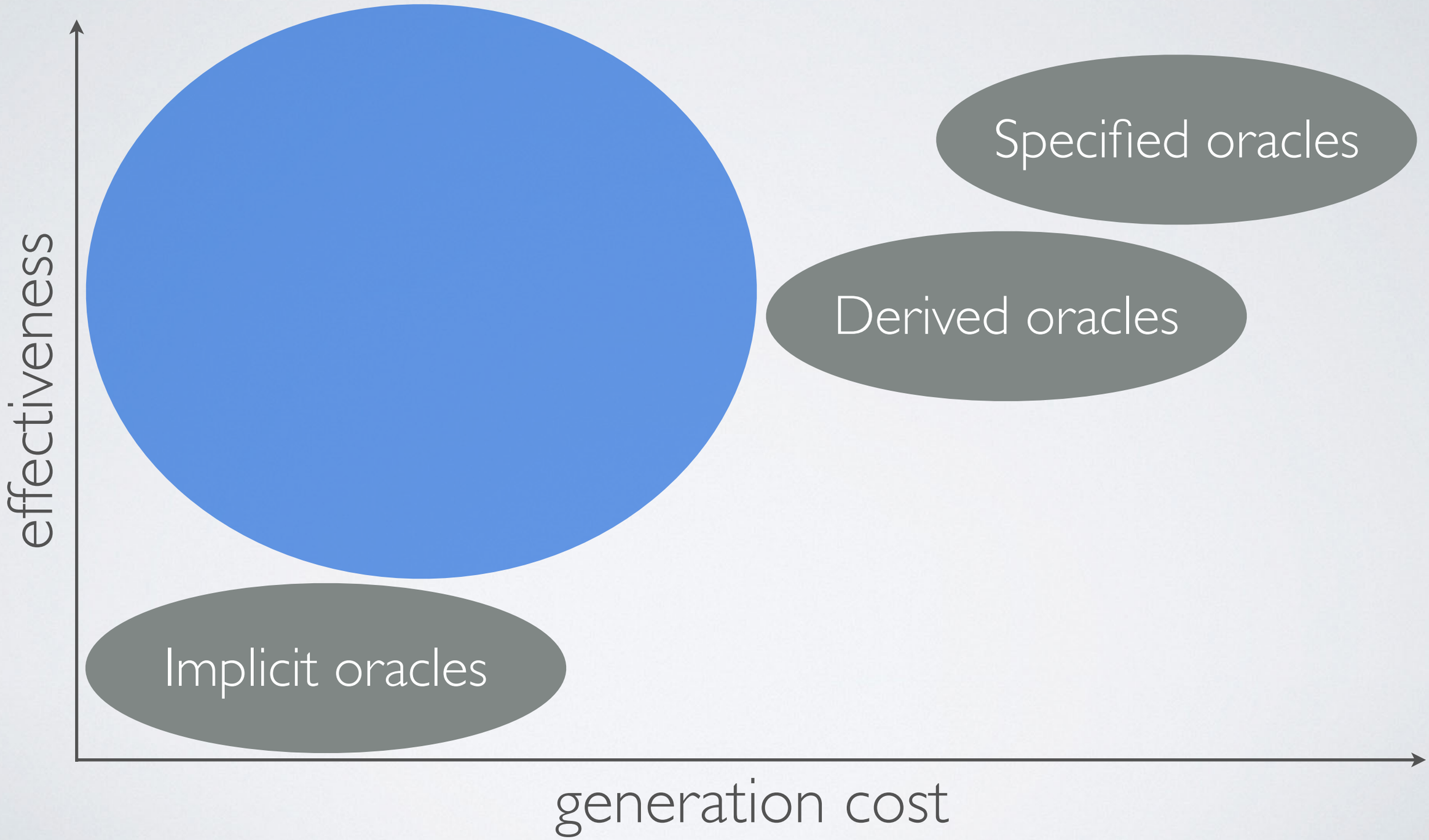
Derived oracles

Implicit oracles

TEST ORACLES



TEST ORACLES



Software systems are **intrinsically redundant**.

We can exploit the intrinsic redundancy to **automatically generate test oracles**.

INTRINSIC REDUNDANCY

Stores a key-value pair in the multimap.

```
public boolean put(K key, V value) {
  Collection<V> collection = map.get(key);
  if (collection == null) {
    collection = createCollection(key);
    if (collection.add(value)) {
      totalSize++;
      map.put(key, collection);
      return true;
    } else {
      throw new AssertionError("...");
    }
  } else if (collection.add(value)) {
    totalSize++;
    return true;
  } else {
    return false;
  }
}
```

Stores a key-value pairs in the multimap with one key and multiple values.

```
public boolean putAll(K key, Iterable<V> values) {
  if (!values.iterator().hasNext()) {
    return false;
  }
  Collection<V> collection =
                              getOrCreateCollection(key);
  int oldSize = collection.size();
  boolean changed = false;
  if (values instanceof Collection) {
    Collection<? extends V> c =
                              Collections2.cast(values);
    changed = collection.addAll(c);
  } else {
    for (V value : values) {
      changed |= collection.add(value);
    }
  }
  totalSize += (collection.size() - oldSize);
  return changed;
}
```

INTRINSIC REDUNDANCY

Stores a key-value pair in the multimap.

```
public boolean put(K key, V value) {
  Collection<V> collection = map.get(key);
  if (collection == null) {
    collection = createCollection(key);
    totalSize++;
    return true;
  } else {
    return false;
  }
}
```

Stores a key-value pairs in the multimap with one key and multiple values.

```
public boolean putAll(K key, Iterable<V> values) {
  if (!values.iterator().hasNext()) {
    return false;
  }
    } else {
      for (V value : values) {
        changed |= collection.add(value);
      }
    }
    totalSize += (collection.size() - oldSize);
    return changed;
  }
```

ArrayListMultimap.put(key, value) ≡
ArrayListMultimap.putAll(key, Arrays.asList(value))

INTRINSIC REDUNDANCY AS TEST ORACLE

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

```
map.put(1, "Bob");
```

≡

```
List<String> values = Arrays.asList("Bob");
map.putAll(1, values);
```

INTRINSIC REDUNDANCY AS TEST ORACLE

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

map = {}

↓

```
map.put(1, "Bob");
```

≡

map = {}

↓

```
List<String> values = Arrays.asList("Bob");
map.putAll(1, values);
```

INTRINSIC REDUNDANCY AS TEST ORACLE

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

INTRINSIC REDUNDANCY AS TEST ORACLE

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

Test case

```
ArrayListMultimap map = ...
map.put(1,“Bob”);
map.remove(1);
```

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

Test case

```
ArrayListMultimap map = ...
map.put(1, "Bob");
map.remove(1);
```

Test case

ArrayListMultimap map = ...

map.put(1,“Bob”);

List<String> values =
Arrays.asList(“Bob”);
map.putAll(1, values);

compare

map.remove(1);

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

Test case

ArrayListMultimap map = ...

map.put(1, “Bob”);

List<String> values = Arrays.asList(“Bob”);
map.putAll(1, values);

compare

map.remove(1);

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

Test case

ArrayListMultimap map = ...

map.put(1, “Bob”);

map.remove(1);

List<String> values = Arrays.asList(“Bob”);
map.putAll(1, values);

compare

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

ArrayListMultimap map = ...

map' = duplicate(map)

ret = map.put(1, "Bob");

List<String> values = Arrays.asList("Bob");
ret' = map'.putAll(1, values);

compare(map, map')
compare(res, res')

map.remove(1);

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

ArrayListMultimap map = ...

customized deep-cloning library

map' = duplicate(map)

ret = map.put(1, "Bob");

List<String> values = Arrays.asList("Bob");
ret' = map'.putAll(1, values);

compare(map, map')
compare(res, res')

map.remove(1);

ArrayListMultimap.put(K, V) ≡ ArrayListMultimap.putAll(K, Iterable)

ArrayListMultimap map = ...

map' = duplicate(map)

customized deep-cloning library

ret = map.put(1, "Bob");

List<String> values = Arrays.asList("Bob");
ret' = map'.putAll(1, values);

map.remove(1);

compare(map, map')
compare(res, res')

hybrid: equals + observational equivalence

EVALUATION

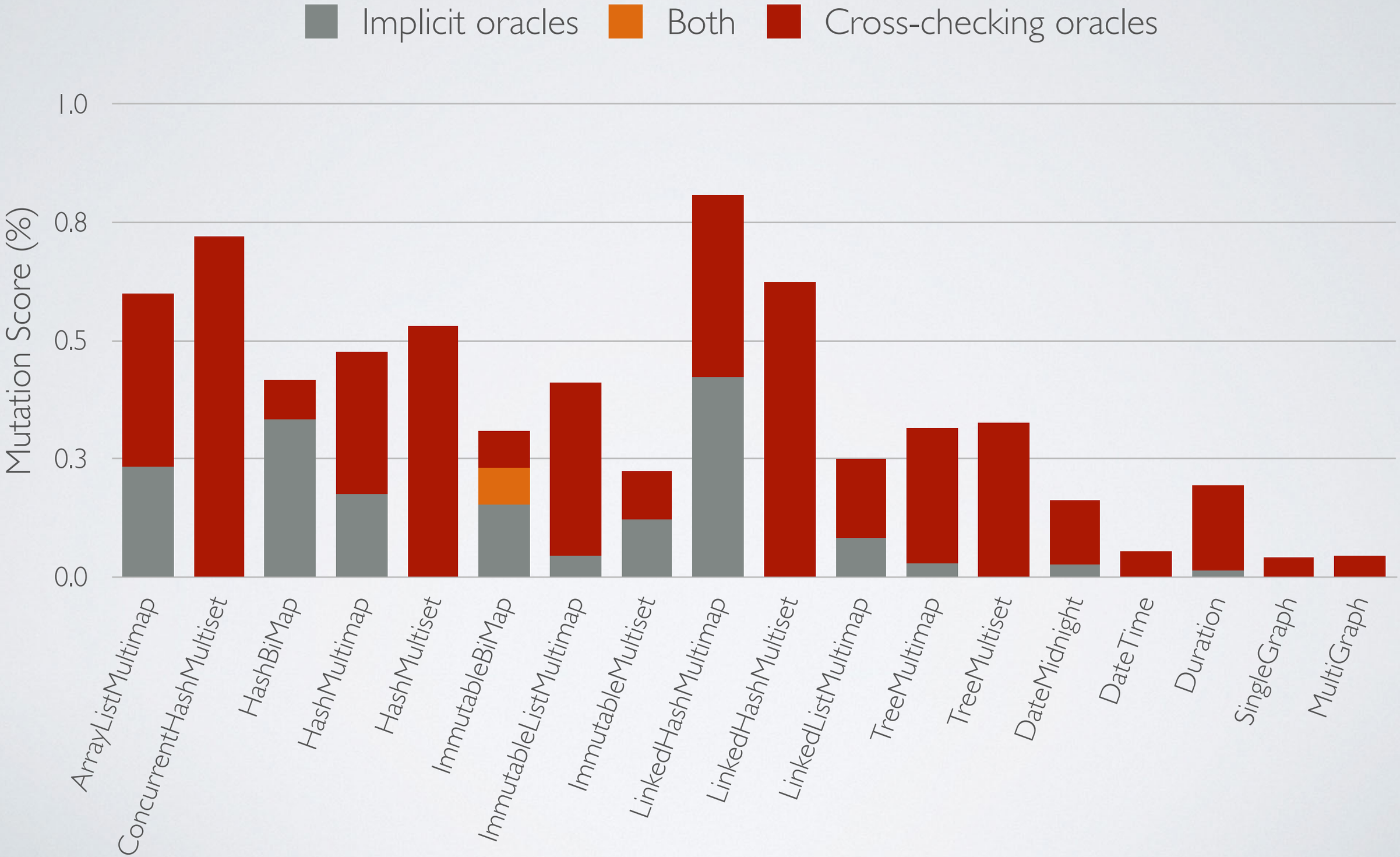
Subject Class		# Methods	# Redundancy Specs.
Guava	ArrayListMultimap	25	29
	ConcurrentHashMultiset	27	32
	HashBiMap	20	16
	HashMultimap	24	29
	HashMultiset	26	30
	ImmutableBimap	25	19
	ImmutableListMultimap	30	34
	ImmutableMultiset	32	50
	LinkedHashMultimap	24	30
	LinkedHashMultiset	26	31
	LinkedListMultimap	24	29
	TreeMultimap	26	28
	TreeMultiset	35	37
JodaTime	DateMidnight	118	20
	DateTime	153	27
	Duration	44	6
GraphStream	SingleGraph	107	41
	MultiGraph	107	41

EVALUATION

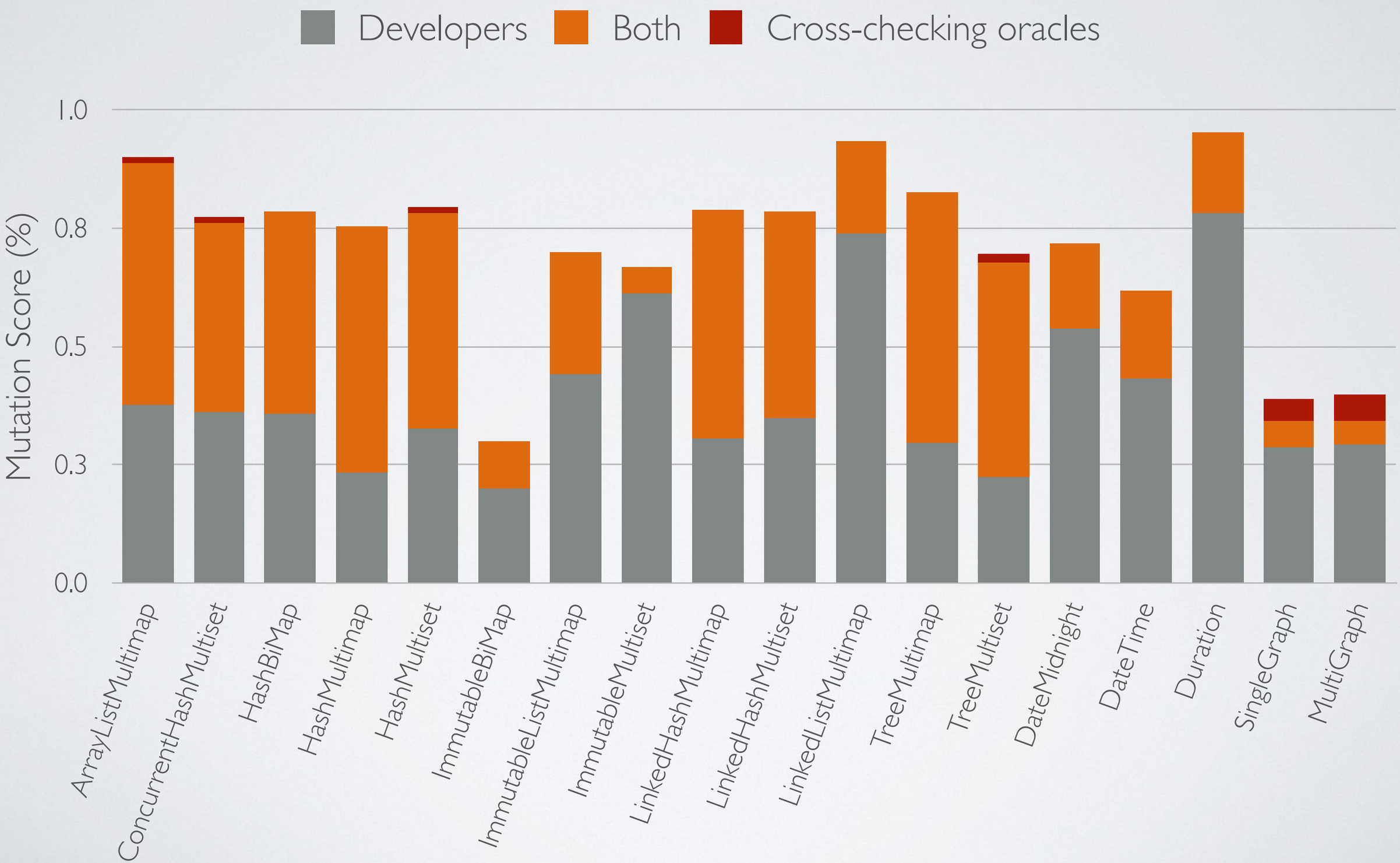
Subject Class		# Methods	# Redundancy Specs.
G	ArrayListMultimap	25	29
	ConcurrentHashMultiset	27	32
	HashBiMap	20	16
	HashMultimap	24	29
	TreeMultiset	35	37
JodaTime	DateMidnight	118	20
	DateTime	153	27
	Duration	44	6
GraphStream	SingleGraph	107	41
	MultiGraph	107	41

3 Java Libraries
18 Classes
873 Methods
529 Equivalent Sequences

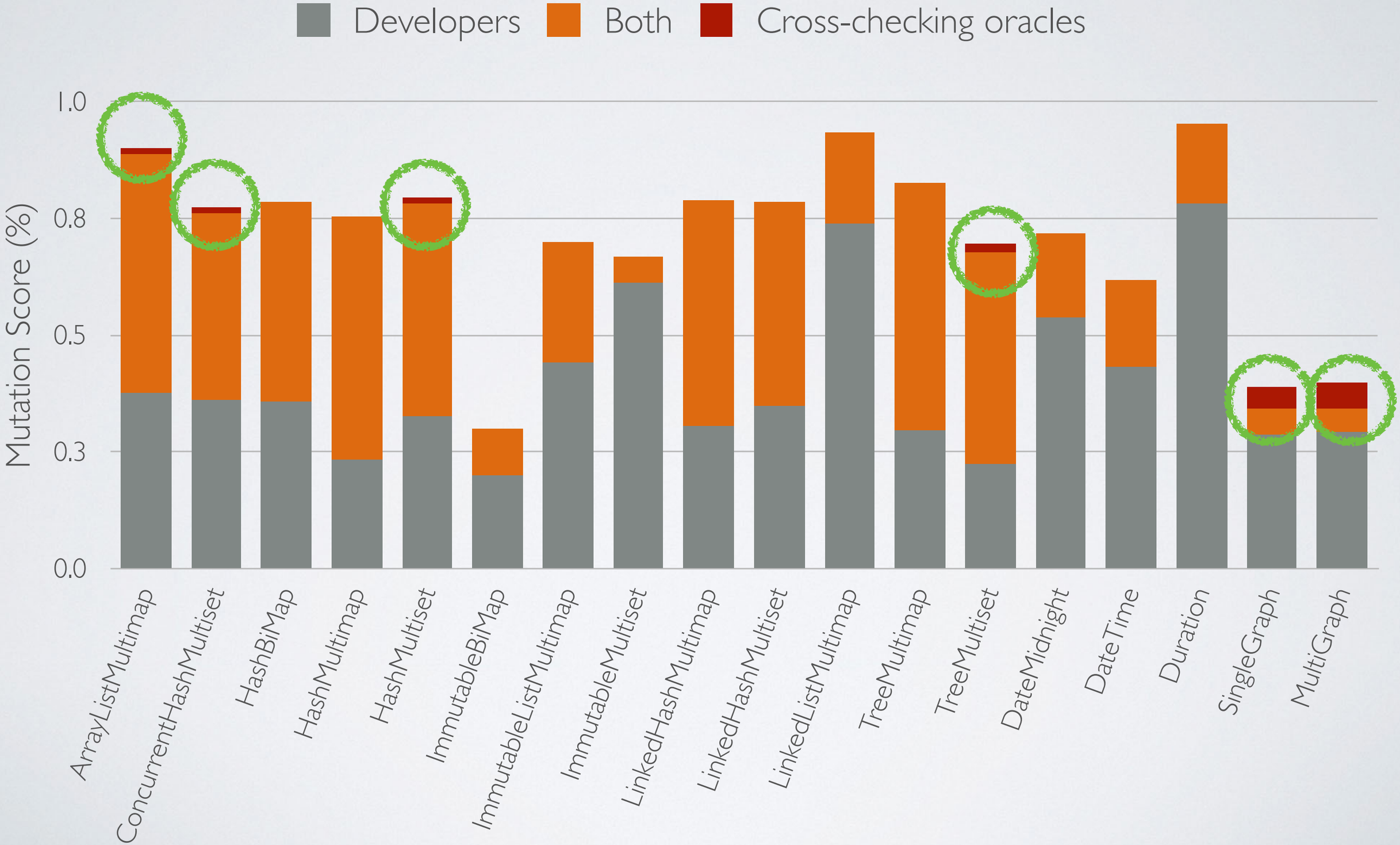
EARLY RESULTS (GENERATED TESTS)



EARLY RESULTS (HAND-WRITTEN TESTS)



EARLY RESULTS (HAND-WRITTEN TESTS)



RESEARCH PLAN

- Oracles execution
- Equivalence check
- Evaluation

RESEARCH PLAN

- Oracles execution
- Equivalence check
- Evaluation

- Study and improve the completeness

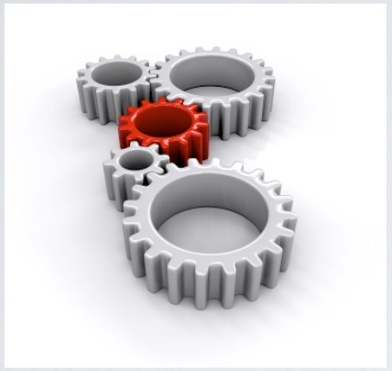
RESEARCH PLAN

- Oracles execution
- Equivalence check
- Evaluation

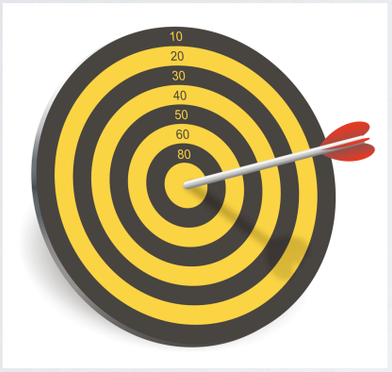
- Study and improve the completeness

- Reduce the cost: Automatic synthesis of equivalent sequences

RESEARCH PLAN

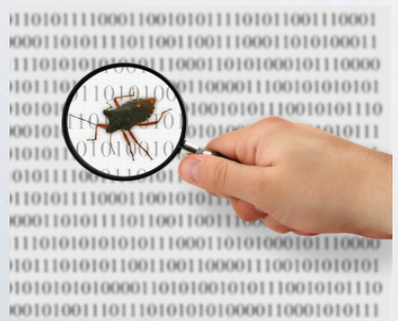


- Oracles execution
- Equivalence check
- Evaluation



- Study and improve the completeness

- Reduce the cost: Automatic synthesis of equivalent sequences



- Fault localization/Fault fixing

Cross-Checking Oracles from Intrinsic Software Redundancy

A. Carzaniga, A.Goffi, A. Gorla, A. Mattavelli, M. Pezzè

June 6th, 11:30, Hall 1

(Research Track, Testing and Conformance Verification)