Mutation Analysis Beyond Quality Estimation

Gordon Fraser, The University of Sheffield

Measure how good a test suite is

Measure how good a test suite is

Measure how good a test suite is

Compare two testing techniques

Measure how good a test suite is

Compare two testing techniques

Identify how to improve test suites

Measure how good a test suite is

Compare two testing techniques

Identify how to improve test suites

Measure how good a test suite is

Compare two testing techniques

Identify how to improve test suites

Drive automated test generation

Measure how good a test suite is

Compare two testing techniques

Identify how to improve test suites

Drive automated test generation

Only test analysis technique that is sensitive to test oracles

Measure how good a test suite is

Compare two testing techniques

Identify how to improve test suites

Drive automated test generation

Only test analysis technique that is sensitive to test oracles

Measure how good a test suite is

Compare two testing techniques

Identify how to improve test suites

Drive automated test generation

Only test analysis technique that is sensitive to test oracles

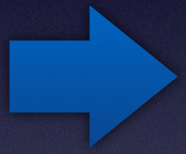
Drive various test-related analyses

Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012

```
class Foo {
  int bar(int x) {
    return 2 * x;
  }
}
```

Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012

```
class Foo {
  int bar(int x) {
    return 2 * x;
  }
}
```



```
void test() {
  f = new Foo();
  y = f.bar(10);

}
```

Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012

```
LocalDate date = new LocalDate(2010, 7, 15);
date.plusYears(1);
```

```
LocalDate date = new LocalDate(2010, 7, 15);
date.plusYears(1);
```

```
LocalDate date = new LocalDate(2010, 7, 15);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2010);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getYear(), 2010);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
```

```
LocalDate date = new LocalDate(2010, 7, 15);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2010);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getYear(), 2010);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
```

```
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2011);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfEra(), ...);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
```

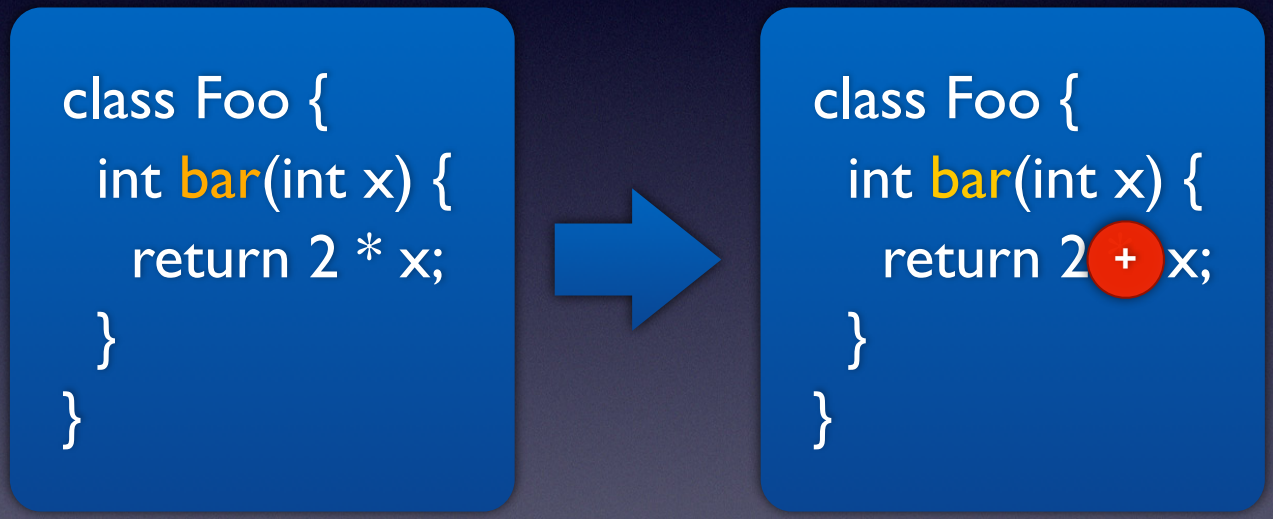
```
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2011);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfEra(), ...);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
```

```
assertEquals(date.getDayOfMonth(), ...);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
assertEquals(date.size(), 3);
assertEquals(date.getValue(YEAR), 2011);
assertEquals(date.getValue(MONTH_OF_YEAR), 7);
assertEquals(date.getValue(DAY_OF_MONTH), 15);
assertEquals(date.getLocalMillis(), ...);
assertEquals(date, date);
assertEquals(date.compareTo(date), 0);
assertEquals(date.getYearOfEra(), ...);
assertEquals(date.getYearOfCentury(), ...);
assertEquals(date.getWeekyear(), ...);
assertEquals(date.getMonthOfYear(), 7);
assertEquals(date.getWeekOfWeekyear(), ...);
assertEquals(date.getDayOfWeek(), ...);
assertEquals(date.getDayOfMonth(), ...);
```

```
LocalDate date = new LocalDate(2010, 7, 15);
date.plusYears(1);
assertEquals(date.getYear(), 2011);
assertEquals(date.getValue(YEAR), 2011);
```

```
class Foo {
  int bar(int x) {
    return 2 * x;
  }
}
```

Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012



Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012

Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012

Fraser, Zeller: "Generating Unit Tests and Oracles", ISSTA 2010/TSE 2012

```
YearMonthDay var0 = new YearMonthDay();
TimeOfDay var1  = new TimeOfDay(var0);
DateTimeZone var2 = DateTimeZone.UTC;
DateTime var3 = var0.toDateTime(var1);
DateTime var4 = var3.withZone(var2);
```

Parameterized Unit Test

```
void test(TimeOfDay var1, DateTimeZone var2,
          YearMonthDay var0) {

    assume(var0 != null);
    assume(var1 != null);
    assume(var2 != null);

    DateTime var3 = var0.toDateTime(var1);
    DateTime var4 = var3.withZone(var2);

    assert(var3.equals(var4));
}
```

```
void concrete_test() {
  Caverphone var0 = new Caverphone();
  String var1 = "EL";
  String var2 = "ILLA";
  boolean var3 = var0.isCaverphoneEqual(var1, var2);
}
```

Fraser, Zeller: "Generating Parameterized Unit Tests", ISSTA 2011

```
void concrete_test() {
  Caverphone var0 = new Caverphone();
  String var1 = "EL";
  String var2 = "ILLA";
  boolean var3 = var0.isCaverphoneEqual(var1, var2);
}
```

```
void parameterized_test(String input1, String input2) {
  assume(!input1.equals(input2));

  Caverphone var0 = new Caverphone();
  boolean var1 = var0.isCaverphoneEqual(input1, input2);

  assert(var1 == false);
}
```

Fraser, Zeller: "Generating Parameterized Unit Tests", ISSTA 2011