Search-Based Test Amplification

COLQ — M2 SIF

Supervisor: Benoit Baudry
KTH, Sweden
Wednesday 7th February, 2018 – Friday 22nd June, 2018

Simon Bihel simon.bihel@ens-rennes.fr

Wednesday 31st January, 2018

University of Rennes I
École Normale Supérieure de Rennes

Introduction

Test Suites

Context:

- Software projects are now accompanied by strong test suites
- Takes time to write
- Still missing some bugs due to focus on nominal paths when writing test cases

Test Suites

Context:

- Software projects are now accompanied by strong test suites
- Takes time to write
- Still missing some bugs due to focus on nominal paths when writing test cases

Related works:

- Measure the quality of test suites
- Automatically write test suites
- Amplify existing test suites

Concepts

System-Under-Test: function, class, whole program…

Inputs	E.g. function parameters, method calls to setup and stimulate an object
Assertions	Used to test whether the function’s output is correct, that the object is in the right state

Test Example

```
testIterationOrder() {
  TreeList tl = new TreeList(10);
  for (int i = 0; i < size; i++) {
    tl.add(i);
  }
  int i = 0;
  ListIterator it = tl.listIterator();
  while (it.hasNext()) {
    Integer val = it.next();
    assertEquals(i++, val.intValue());
  }
}
```

Metrics for Test Suites

Goal

Detect parts that are not tested.

Metrics for Test Suites

Goal

Detect parts that are not tested.

Code Coverage

Number of instructions or branches executed by the test suite.

Metrics for Test Suites

Goal

Detect parts that are not tested.

Code Coverage

Number of instructions or branches executed by the test suite.

Mutation Score

1. Create *mutants* (i.e. bugged versions) of the main software (e.g. change a > with a <=).
2. Count how many mutants for which the test suite fail.

Automated Test Generation

Goal

Generate tests from scratch to fulfill a given metric.

Large search space of instructions and values.

Search-based techniques[1]

Random, iterative and heuristic-based techniques (e.g. Genetic Algorithms, simulated annealing).

[1] McMinn, “Search-based software testing: Past, present and future”, 2011; Fraser and Arcuri, “Evosuite: automatic test suite generation for object-oriented software”, 2011.

[2] Barr et al., “The oracle problem in software testing: A survey”, 2015.

Automated Test Generation

Goal

Generate tests from scratch to fulfill a given metric.

Large search space of instructions and values.

Search-based techniques[1]

Random, iterative and heuristic-based techniques (e.g. Genetic Algorithms, simulated annealing).

The oracle problem[2]

What should the output of a test be?

- Avoid this by focusing on regression testing.

[1] McMinn, “Search-based software testing: Past, present and future”, 2011; Fraser and Arcuri, “Evosuite: automatic test suite generation for object-oriented software”, 2011.

[2] Barr et al., “The oracle problem in software testing: A survey”, 2015.

Test Suite Amplification

Motivations[3]

- Reduce search-space by using the existing test suite as a (good) starting population.
- Use knowledge in hand-written tests for a better oracle.

[3] Danglot et al., “The Emerging Field of Test Amplification: A Survey”, 2017.

Test Data Regeneration[4]

Goal

Modify a test case while keeping the same branch coverage.

- Avoids over-fitting.
- Helps for fault detection.

Test cases are vectors of integers. Modified tests are called *neighbors*.

[4] Yoo and Harman, “Test data regeneration: generating new test data from existing test data”, 2012.

Test Data Regeneration

Modification *operator*:

- $+1$ and -1
- $*2$ and $\lceil /2 \rceil$

Test Data Regeneration

Modification *operator*:

- $+1$ and -1
- $*2$ and $\lceil /2 \rceil$

Reduces the search space with a *Search Radius*:

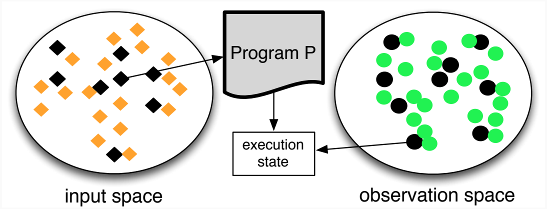
- Move further away from original test.
- Limit the number of modifications to avoid searching too far.

Experiments needed to find best values.

DSpot[5]

Goal

Create tests for undetected mutants.

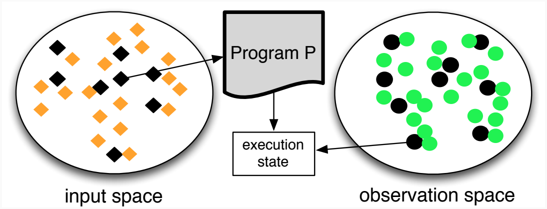


[5]Baudry et al., “DSpot: Test Amplification for Automatic Assessment of Computational Diversity”, 2015.

DSpot[5]

Goal

Create tests for undetected mutants.



New tests ought to be approved by the developers. Do not mess with people's code.

[5] Baudry et al., "DSpot: Test Amplification for Automatic Assessment of Computational Diversity", 2015.

DSpot — Amplification Operators

Input amplification

Literals → replaced with neighbor values.

Method calls → duplicated, removed or made-up (with random or default parameters).

DSpot — Amplification Operators

Input amplification

Literals → replaced with neighbor values.

Method calls → duplicated, removed or made-up (with random or default parameters).

Only one operator can be applied

DSpot — Amplification Operators

Input amplification

Literals → replaced with neighbor values.

Method calls → duplicated, removed or made-up (with random or default parameters).

Only one operator can be applied

Assertion amplification

Capture the state of the system after the test's execution.

Planned Work

Easing the Review Process

Human review process

- Takes time.
- Misunderstood tests are ignored or labelled as false-positive.

What we could do

- Add explanations → easier to understand what the target of the new test is.
- Avoid testing completely different things.
- Order tests by importance.

Efficient Generation

Minimal set of tests **if** each test stays simple and logical.

Stacking operators

- More properties covered.
- Increases the search space, study the usefulness of each operator.

Modify existing test case or add a new one?

Conclusion

Summary

1. Practitioners need help designing good quality test suites.
2. Search spaces are vast and tests need to be precise.
3. Automated techniques are available to enhance hand-written test.
4. The internship will focus on adapting these techniques for human interactions needs.

Expected contribution: **explanations** and **focus**.

EvoSuite

- Test Suite Generation *from scratch*.
- State-of-the-Art & industry grade.

Differences

- Treats test suites as a whole.
- Gen. Algo. with tests cases as genes.

What could be adapted

- Tests minimization.