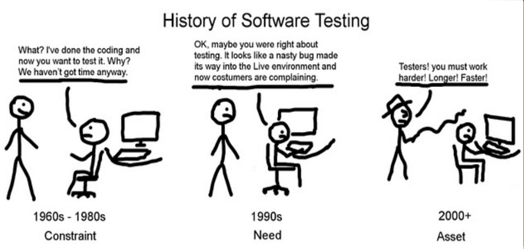
Metamorphic testing

Huai Liu
huai.liu@vu.edu.au

Outline

- Testing
- Test oracle & oracle problem
- Metamorphic testing & metamorphic relations
- State of art
- Future work

Testing



Testing

- IEEE definition:
  Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item.

Validation & Verification

- Validation
  - Are we building the right product?
  - The software should do what the user really requires
- Verification
  - Are we building the product right?
  - The software should conform to its specification

Error, fault, and failure

- Error
- Fault
- Failure

Error, fault, and failure

- When developing software, people make errors, these become faults in the software which then manifest themselves as failures when the software is run.
- One error may lead to several different faults, each of which in turn leads to several different failures.

Error, fault, and failure

- IEEE defines “error” as:
  - The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.
  - An incorrect step, process, or data definition.
  - An incorrect result.
  - A human action that produces an incorrect result.

Error, fault, and failure

- But: While all four definitions are commonly used, one distinction assigns
  - definition 1 to the word “error”,
  - definition 2 to the word “fault”,
  - definition 3 to the word “failure”, and
  - definition 4 to the word “mistake”.

Error, fault, and failure

- Fault
  - A defect in a hardware device or component.
  - An incorrect step, process, or data definition in a computer program.
- Failure
  - The inability of a system or component to perform its required functions within specified performance requirements.

Error, fault, and failure

- Code to decide whether a triangle is right-angled.
  - if (a * a + b * b == c * c)
    cout << “right-angled” << endl;
  - if (a * a * a + b * b * b == c * c * c)
    cout << “right-angled” << endl;

Error, fault, and failure

- Error: the programmer uses a wrong formula
- Faults: a * a → a * a * a; b * b → b * b * b; c * c → c * c * c.
- Failures: some triangles are mistakenly considered as non-right-angled; while some triangles are mistakenly considered a right-angled.

Error, fault, and failure

- One error may lead to several different faults, each of which in turn leads to several different failures.
- What is the main target of testing?
  - demonstrates the presence of fault.
  - does not demonstrate the absence of fault.

What is testing?

- Testing: by experiment,
  - find faults in software
  - establish quality of software
- A successful test:
  - finds at least one fault
  - gives quality judgement with maximum confidence with minimum effort

What is testing?

Fundamental problems

- Reliable test set problem
  - exhaustive testing?
  - reliable for any given program?
- Oracle problem
  - test oracle
  - difficult to verify test results

Fundamental problems

Fundamental problems

- Infinite monkey theorem
  - A monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type a given text, such as the complete works of William Shakespeare.
- Unfortunately, we, as testers, are never given an infinite amount of time!

Fundamental problems

- p : a program
- D : the input domain (all possible inputs to p)
- If we test p on every element of D, and the results are correct, then p is a correct program
- not practical when D has a large or infinite size.

Fundamental problems

- We want to find a subset of test cases T from D
  - If $p(T)$ is correct, then $p(D)$ is correct
  - We say T is a *Reliable Test Set*.

Fundamental problems

- Problem: In general, T (of finite size) for any given program cannot be constructed effectively, unless T = D (known as "exhaustive testing")
- In other words: reliable test sets are not attainable in general
- As a result, testing cannot prove program correctness in most situations.

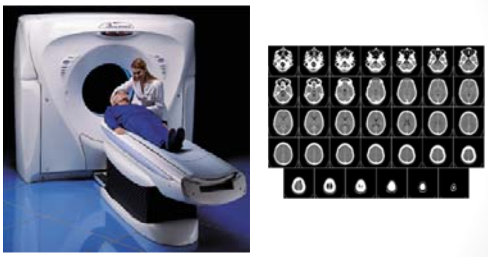
Fundamental problems

- Test oracle
  - Expected output
  - A way to check the correctness of program output
- $ax^2 + bx + c = 0$
  - $x = [-b \pm (b^2 - 4ac)^{1/2}] / 2a$
  - Replace x by the calculated results

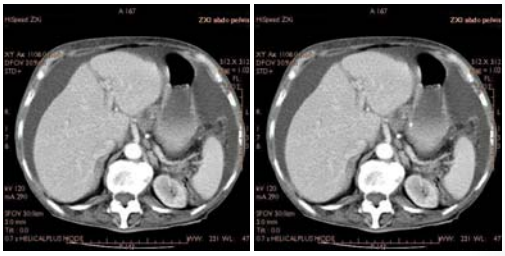
Fundamental problems

- Oracle does not always exist
  - It is too expensive to apply oracle in practice
  - Oracle problem affects the testing effectiveness

Example – Medical image



Example – Medical image



25

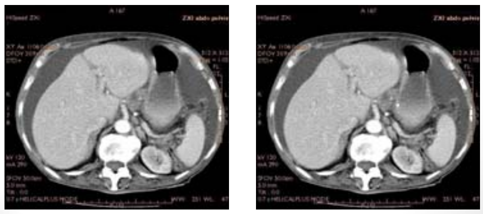
Metamorphic Testing

- Property-based testing
- Alleviate oracle problem
- Verify test results against certain relations
- Multiple executions

27

Metamorphic Testing

- Example
  - Is what we see is correct?
  - Mould? Another machine? Open it?
  - Moving the patient's body



28

Metamorphic testing

- Identify metamorphic relations (MRs)
- Generate source test case
- Construct follow-up test case
  - Based on source test case
  - According to MRs
- Execute test cases
- Verify test results based on MRs

30

Metamorphic testing

- Example
  - MR: the relative position of the white dot to the body won't change.
  - Source test case: scan once
  - Follow-up test case: move the body to left/right, scan again
  - Execution: (x_1, y_1), (x_2, y_2)
  - Verification: $x_1 = x_2$ & $y_1 = y_2$?

Metamorphic relation

- Key part of metamorphic testing
  - Test case generation
  - Test result verification
- Identification of MRs
  - Specification
  - Human intelligence

State of the art

- In the context of testing
  - Bioinformatics
  - Web services
  - Compilers
  - Cybersecurity
  - Self-driving cars

Testing of compilers

- One simple MR
  - If source programs SP and SP′ are equivalent for input I, then their object programs OP and OP′, respectively, are also equivalent on I.
- Straightforward implementation
  - construct SP′ from SP by removing the statements in SP that are not executed with input I.
- Surprisingly good results
  - 100 faults in the popular GCC and LLVM C compilers
  - 50 bugs in OpenCL compilers

Testing of self-driving cars

- Transformation of inputs
  - DNN-UMV: simple image transformation: linear, affine, convolutional
- Checking relationship among outputs
  - Steering angle should be similar after image transformation.
- Lessons and improvements
  - more than 1,000 erroneous behaviors
  - Accuracy improved by up to 46% by retraining DNNs

Testing of self-driving cars

1.1 original

1.2 with added rain

State of the art

- Beyond the context of testing
  - Semi-proving
  - Validation & Verification
  - Quality assessment

V&V framework

- Classifiers
  - Order of the class labels would not affect the final classification
  - kNN algorithm did not have this property.
- Search engines
  - User-oriented
  - Performance degradation when searching large domains

Search engines

State of the art

- Integration of MT with other technologies
  - With fault-based testing
  - With spectrum-based fault localisation
  - With program slicing
  - With fault tolerance

Integration

- The correspondence
  - between a single test case and a group of metamorphic test cases
  - between the pass/fail outcome of a test case and the satisfaction/violation of an MR for the test group

State of the art

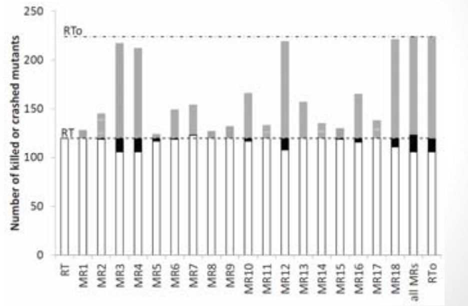
- Evaluation of MRs
  - Fault-detection effectiveness
  - Code coverage

An empirical study

- Subjects
  - 5 programs
  - Thousands of mutants
- Participants
  - 4 universities
  - 4-7 students for each program
- Metamorphic relations
  - 16 to 27 for each program
  - 4 to 6 for each tester

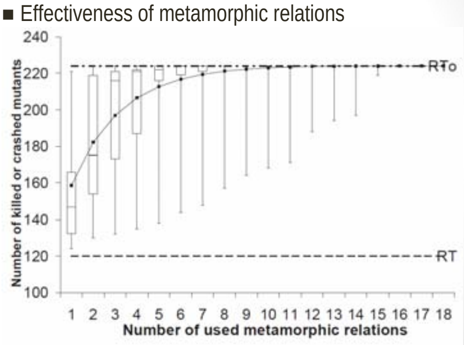
An empirical study

- Effectiveness of metamorphic relations



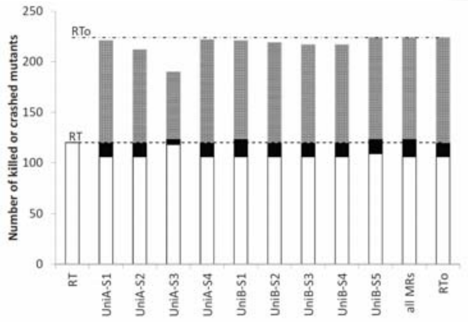
An empirical study

- Effectiveness of metamorphic relations



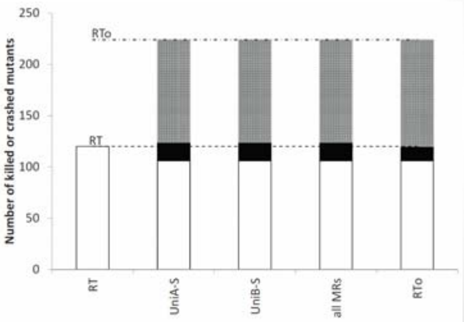
An empirical study

- Capability of tester



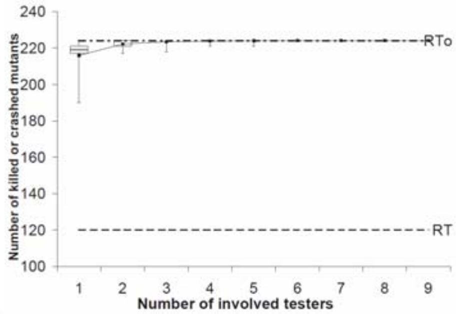
An empirical study

- Capability of tester



An empirical study

- Capability of tester



Challenges & Opportunities

- Wider application of MT
  - Artificial intelligence
  - Image processing
  - Industry patents

Artificial intelligence

- Deep learning for cell imaging
  - Over 6500 diffraction images of three categories of images: normal cells, ghost cell bodies, debris
  - Duplicate 10% of images of each category in the training data set should not affect the classification accuracy
  - Remove one category of the data from the data set should not affect the classification accuracy of the remaining categories

GraphicsFuzz

- Combination of fuzzing and metamorphic testing
  - Highly automatic method for testing graphics drivers
  - Has been successful at exposing a large number of graphics driver defects across a wide range of mobile and desktop platforms

Device (GPU vendor)	Rendering issues	Crashes	Nondeter--ministic renderings	Compile failures	Timeouts	Memouts	Total issues
Huawei Honor 10 (**ARM**)	0	9	0	1	0	0	10
Apple iPhone 6 (**Apple**)	5	9	1	0	0	0	15
Apple iPhone 7 (**Apple**)	5	9	1	0	0	0	15
Apple iPhone 8 (**Apple**)	17	11	0	0	0	0	28
Apple iPhone X (**Apple**)	17	11	0	0	0	0	28
Google Pixel 1 XL (**Qualcomm**)	6	31	0	12	1	0	50

Accenture

RELEVANT PATENTS

Accenture's patents and patent-pending innovations create transparency in machine decision-making

VERIFYING MACHINE LEARNING THROUGH METAMORPHIC TESTING

A typical machine learning (ML) application has an extremely large number of scenarios to test, which can make testing ML applications extremely expensive. This patent presents a new approach to testing ML-based applications through the concept of Metamorphic Testing. Our methodology, based on the underlying mathematical principles of ML algorithms, needs only a few test cases (or even just one) to identify bugs in ML applications, thereby reducing the cost of testing significantly. (Inventors: Anurag Dwarakanath, Manish Ahuja, Samarth Sikand, Raghotham M Rao, R.P. Jagadeesh Chandra Bose, Neville Dubash, Sanjay Podder, Kishore Durg)

Challenges & Opportunities

- Identification of MRs
  - Ad hoc
  - From existing MRs
  - Systematic Construction

Challenges & Opportunities

- Other areas
  - Quality measurement
  - Cloud & crowd
  - Big data
  - Agile development

System understanding & use

(a) Top American inventors returned by Google, under the category "Inventors > United States."

(b) An equivalent query term in Chinese yielded a very different list of American inventors, still under the category "Inventors > United States."

55

System understanding & use

(a) Walking navigation in London: 3 ft, 1 min.

(b) Bad case detected: MR violation after reversing origin and destination: 0.5 miles, 10 min.

56

System understanding & use

(c) Walking navigation in Hong Kong: 250 m, 3 min.

(d) Bad case detected: MR violation after reversing origin and destination: 750 m, 10 min.

57

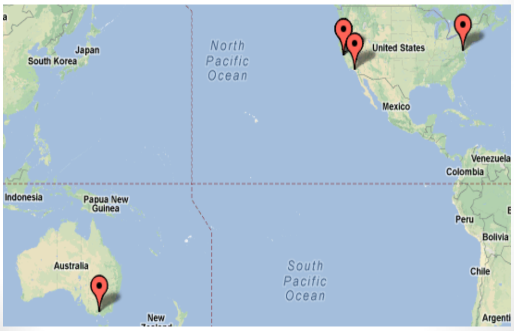
System understanding & use

(a) The same bad case as in Fig. 11b but with a slightly different starting point.

(b) The bad case was detected and the shortcut was found by swapping the origin and destination, using the same "change direction" MR.

58

Example – Travel planning



59

Metamorphic Testing

- Example
  - Have I got the best deal?
  - Ask another quote? Trust it anyway?
  - Changing the search criteria

61

Metamorphic testing

- Example
  - MR: changing stopover but keep starting and ending points unchanged won't give a better deal.
  - Source test case: Melbourne → Washington → Melbourne
  - Follow-up test case: Melbourne → SF → Washington → LA → Melbourne
  - Execution: $price_d$, $price_s$
  - Verification: $price_d \leqslant price_s$?

63

References

- Chen et al. "Metamorphic Testing: A Review of Challenges and Opportunities." ACM Computing Surveys
- Liu et al. "How Effectively does Metamorphic Testing Alleviate the Oracle Problem?" IEEE Transactions on Software Engineering
- Sun et al. "Metamorphic Testing for Web Services: Framework and a Case Study." International Conference on Web Services
- Zhou et al. "Metamorphic Testing for Software Quality Assessment: A Study of Search Engines." IEEE Transactions on Software Engineering

64

References

- Segura et al. "Metamorphic Testing of RESTful Web APIs." IEEE Transactions on Software Engineering
- Segura et al. "A Survey on Metamorphic Testing." IEEE Transactions on Software Engineering
- Tian et al. "DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars." International Conference on Software Engineering
- Chen et al. "Metamorphic Testing for Cybersecurity." IEEE Computer

65

More to come

- Zhou et al. "Metamorphic Relations for Enhancing System Understanding and Use." IEEE Transactions on Software Engineering
- Zhou et al. A work in testing self-driving cars. Communications of the ACM
- Sun et al. METRIC*. IEEE Transactions on Software Engineering
- Patents

66


Thank You!
67