Metamorphic Testing

Tsong Yueh Chen
Swinburne University of Technology

20th CREST Open Workshop

Oracle Problem

- Oracle - a mechanism to verify the correctness of the outputs
- Oracle Problem – absence of oracle or too expensive to apply the oracle

A Motivating Example

- *sin* function
  - $sin(0^o)=0$
  - $sin(30^o)=0.5$
- Suppose the program returns:

  $sin(29.8^o)=0.51234$ incorrect

  $sin(29.8^o)=0.49876$ correct?

A Motivating Example

- *sin* function has the following properties
  - *sin*(x)= *sin*(x+360)
  - ……
- Execute the program with *sin*(389.8^{o})

  compare *sin*(29.8^{o}) and *sin*(389.8^{o})

Metamorphic Testing (A Simplified Form)

- Source (original) test cases – generated according to certain strategies
- Follow-up test cases could be constructed from the successful test cases with reference to certain properties of the problem
- Such properties are known as metamorphic relations

Metamorphic Testing

- Some Characteristics
  - Necessary properties of the problem – not restricted to identity relations and numeric relations
  - Multiple executions – at least one source test case and one follow-up test case
  - Follow-up test cases may depend on the outputs of the source test cases

Metamorphic Testing

- test case – metamorphic test group
- test outcome of pass or failure – test outcome of satisfaction or violation of a metamorphic relation

Metamorphic Testing

- Aimed at alleviating the oracle problem
- A property-based testing method – providing a new perspective to design test cases
- Metamorphic relation

Categories of Research in MT

- Use of MT to test application domains which have oracle problem
- Integration of MT with other software analysis and testing methods
- Its own theory

Testing Software with Oracle Problem

- Bioinformatics programs
- Embedding systems
- Machine learning software
- Optimization systems
- …..

Integration with Other Methods

- Slicing – metamorphic slice
- Spectrum Based Fault Localizations (SBFL)
- Symbolic execution – semi-proving which supports debugging, testing and proving

Metamorphic Slices

- Slicing may be the most *important* concept in software analysis
- Conventional slices are data-based
- Metamorphic slices are also property-based
- Many applications of slicing assume the existence of a test oracle

Existing SBFL Techniques

- A test suite – the test outcome of each test case is known (oracle assumed)
- A program spectrum – execution slice
- A risk evaluation formula to assign a risk value of being faulty to each statement

SBFL Without Oracle

- slice – metamorphic slice
- test case – metamorphic test group
- pass or failure of a test case – satisfaction or violation of a metamorphic test group

Integration with Symbol Execution

- Proving metamorphic relations – metamorphic proving
- Sometimes able to prove program
- Providing useful information to debug

Theory for Metamorphic Testing

- Metamorphic Relations
  - Necessary properties involving multiple inputs
    - Identification or generation of MRs
  - Many MRs
    - Set of MRs treated as input domain – selection strategies for MRs
    - Prioritization of MRs

Generation of MRs

- Is it feasible to identify or generate MRs?

Generation of MRs

- Should target at
  - Development of guidelines or systematic methods for a specific type of application domains
  - Development of semi-automated methods

Prioritization of MRs

Consider *sin*(x)

MR1: $sin(x) = sin(x + 2\pi)$

MR2: $sin(x) = -sin(x + \pi)$

MR3: $sin(-x) = -sin(x)$

MR4: $sin(x) = sin(\pi - x)$

MR5: $sin(x) = -sin(2\pi - x)$

…

Priorization Approaches

- Usage profile
- Algorithm

Usage Profile

- $sin(x)$
  - Electrical engineers
    - $sin(x) = sin(x + 2\pi)$
  - Surveyors
    - $sin(-x) = -sin(x)$
    - $sin(x) = sin(\pi - x)$

Algorithm

- A problem may be solved by more than one algorithm – sorting, adaptive random testing
- The same algorithm may be implemented in different ways

Example

- Shortest Path problem:

  $SP(G, a, b)$ using forward expansion

- $|SP(G, a, b)| = |SP(G, a, c)| + |SP(G, c, b)|$
- $|SP(G, a, b)| = |SP(G, b, a)|$

A Test Case Selection Strategy

- Observation: MT reveals bugs in some software that has been used and tested by conventional testing methods for a long time.

  schedule and print_token
- A testing method for end-users

End-User Software Engineering

- Source test case selection strategy – any available; otherwise special values, random or ad hoc selection
- Selection of MRs –
  - usage profile
  - end-user's domain knowledge
  - coding

Specifications

Is MT a Black-Box Method?

Example

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} -$$

Example

- MR1 $sin(-x) = -sin(x)$
- MR2: $sin(x) = sin(x + 2\pi)$

Conclusion

Thanks