Metamorphic Testing: A Simple Method for Testing Non-Testable Programs

Tsong Yueh Chen

tychen@swin.edu.au
Swinburne University of Technology
Australia

Test Oracle

- A mechanism or procedure against which the computed outputs could be verified

Example

To find the roots of the following polynomial

$x**100 + 3*(x**99) - x**98 + \ldots\ldots + 5$

Suppose the solutions for x are: 2.0, 6.5, ..

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?

Example

- Shortest path problem $SP(G, a, b)$
- Suppose the program returns:
  - $|SP(G, a, a)| = 5$ incorrect
  - $|SP(G, a, b)| = 10$ where a and b are neighbours
  - $|SP(G, a, b)| = 1{,}000{,}001$ correct or incorrect?

Non-Testable Programs

- Absence of test oracle
- Too expensive to apply test oracle

Intuition of Metamorphic Testing

Though we do not know the correctness of the output of any individual input

We may know the relation between some related inputs and their outputs

Example

- Suppose *sin*(29.8^{o}) returns 0.49876
- *sin* function has the following property
  - *sin*(x)= *sin*(x+360)
- Compute 29.8^{o} + 360^{o} = 389.8^{o}
- Execute the program with 389.8^{o} as an input
- Check whether *sin*(29.8^{o}) = *sin*(389.8^{o})

Metamorphic Testing (A Simplified Form)

- Define source (initial) test cases using some test case selection strategies
- Identify some properties of the problem (referred to as the metamorphic relations)
- Construct follow-up test cases from the source test cases with reference to the identified metamorphic relations
- Verify the metamorphic relations

Categories of Research in MT

- Applications of MT to specific application domains with oracle problem
- Integration of MT with other software analysis and testing methods
- Theory for MT

Applications of MT

Successful Applications of MT

- Bioinformatics programs
- Embedded systems
- Machine learning software
- Optimization systems
- Compilers
- Simulation systems
- ….

Interesting Results

Reveal undetected faults

- Siemens suite
  - print_token, schedule, and schedule_2
- Compilers
- Machine learning tool – Weka
- ……..

A Recent Paper

- Compiler Validation via Equivalence Modulo Inputs, V. Le, M. Afshari and Z. Su, Proceedings of 35th ACM SIGPLAN Conference on Programming Language Design & Implementation (PLDI ’14), 216–226, 2014.

Best Paper Award

Testing Compilers

Their testing method is a MT method

Its MR is:

If programs P and P' are equivalent with respect to input I, then their object codes are equivalent with respect to I.

http://blog.regehr.org.archives/1161

a new test case selection method!

Three Recent Papers

- Metamorphic Model-Based Testing Applied on NASA DAT –an Experience Report, M. Lindvall, D. Ganesan, R. Ardal and R. E. Wiegand, ICSE 2015, 129-138.
- Research Directions for Engineering Big Data Analytics Software, C. E. Otero and A. Peter, IEEE Intelligent Systems, 14-19, January/February 2015.
- A Methodology for Validating Cloud Models Using Metamorphic Testing, A. Nunez and R. M. Hierons, Annals of Telecommunications, Vol. 70(3), 127-135, 2015.

Integration with Other Methods

Example: Spectrum Based Fault Localization

- A statistical approach to predict how likely a program entity is faulty
- Intuition
  - More likely to be faulty if executed by more failed test cases
  - More likely to be faulty if executed by less passed test cases

SBFL

- Use a set of test cases with
  - testing results of pass or fail
  - coverage information whether a program entity is executed or not
- Use a formula to predict how likely a program entity is faulty

Example

$$TS:\begin{pmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{pmatrix}$$

$$P:\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \quad MS:\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$RE:\begin{pmatrix} p & p & p & p & f & f \end{pmatrix}$$

Example

$$TS:\begin{pmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{pmatrix} \qquad A^i:< a^i_{ef} \quad a^i_{ep} \quad a^i_{nf} \quad a^i_{np} >$$

$$P:\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \quad MS:\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad MA:\begin{pmatrix} 2 & 4 & 0 & 0 \\ 0 & 3 & 2 & 1 \\ 0 & 1 & 2 & 3 \\ 1 & 3 & 1 & 1 \\ 2 & 3 & 0 & 1 \\ 2 & 4 & 0 & 0 \end{pmatrix}$$

$$RE:\begin{pmatrix} p & p & p & p & f & f \end{pmatrix}$$

- Tarantula

$$R_T(s_i) = \frac{a_{ef}^i}{a_{ef}^i + a_{nf}^i} / \left(\frac{a_{ef}^i}{a_{ef}^i + a_{nf}^i} + \frac{a_{ep}^i}{a_{ep}^i + a_{np}^i}\right)$$

- Risk values

$$\begin{pmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{2}{5} & \frac{4}{7} & \frac{1}{2} \end{pmatrix}$$

- Ranking list

$$< s_5 \quad s_4 \quad s_1 \quad s_6 \quad s_2 \quad s_3 >$$

SBFL

An Open Problem:

How to apply SBFL on non-testable programs?

Integration of SBFL with MT

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

Integration of SBFL with MT

- Intuition
  - More likely to be faulty if executed by more violated metamorphic test groups
  - More likely to be faulty if executed by less non-violated metamorphic test groups

Example

$$MTS : \begin{pmatrix} g_1 & g_2 & g_3 & g_4 & g_5 & g_6 \end{pmatrix}$$

$$P : \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \quad MS : \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$RE : \begin{pmatrix} n & v & n & n & v & v \end{pmatrix}$$

Example

$$MTS:\begin{pmatrix} g_1 & g_2 & g_3 & g_4 & g_5 & g_6 \end{pmatrix} \qquad A^i:< a^i_{ef} \quad a^i_{ep} \quad a^i_{nf} \quad a^i_{np} >$$

$$P:\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{pmatrix} \quad MS:\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad MA:\begin{pmatrix} 3 & 3 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 0 & 1 & 3 & 2 \\ 2 & 3 & 1 & 0 \\ 3 & 2 & 0 & 1 \\ 3 & 3 & 0 & 0 \end{pmatrix}$$

$$RE:\begin{pmatrix} n & v & n & n & v & v \end{pmatrix}$$

- Tarantula

$$R_T(s_i) = \frac{a_{ef}^i}{a_{ef}^i + a_{nf}^i} / \left(\frac{a_{ef}^i}{a_{ef}^i + a_{nf}^i} + \frac{a_{ep}^i}{a_{ep}^i + a_{np}^i}\right)$$

- Risk values

$$\begin{pmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{4} & 0 & \frac{2}{5} & \frac{3}{5} & \frac{1}{2} \end{pmatrix}$$

- Ranking list

$$< s_5 \quad s_1 \quad s_6 \quad s_4 \quad s_2 \quad s_3 >$$

Other Successful Integrations

One Recent Paper

- A Methodology for Validating Cloud Models Using Metamorphic Testing, A. Nunez and R. M. Hierons, Annals of Telecommunications, Vol. 70(3), 127-135, 2015.

Theory for Metamorphic Testing

Metamorphic Testing

- Some reminders
  - MRs not restricted to identity relations and numeric relations
  - Multiple executions
  - Follow-up test cases may depend on the outputs of the source test cases
  - MT is applicable even if test oracle exists

Metamorphic Relations

Metamorphic Relations

- Identification of MRs
- Prioritization of MRs
- Fault Detection Effectiveness of MRs

Identification of MRs

- MT can be automated except the identification of MRs

Identification of MRs

- Is it feasible to identify or generate MRs?

A Simple and Intuitive Approach

- Select an input
- Modify it, hopefully that the relevant change of output will be somehow predictable.

  If yes, any generalisation?

  If yes, then identify an MR

Identification of MRs

Various approaches

- Machine learning (Columbia; Colorado State)
- Data mutation (Oxford Brookes)
- Coding (Peking)
- Composition (Swinburne)
- Category-choice framework (HK Poly; Wuhan)
- ……
- ……..

Generation by Composition

- Generation of new MRs from existing MRs by composition

Example

- Shortest path problem: $SP(G, a, b)$
- Suppose we have the following MRs
  - MR_A: $|SP(G, a, b)| = |SP(G, b, a)|$.
  - MR_B: $|SP(G, a, b)| = |SP(G', a', b')|$.
- By composition, a new MR is defined as
  - MR_{AB}: $|SP(G, a, b)| = |SP(G', b', a')|$.

Prioritization of MRs

Consider *sin*(x)

MR1: *sin*(x) = *sin*(x + 2 π)

MR2: *sin*(x) = -*sin*(x + π)

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

MR4: *sin*(x) = *sin*(π-x)

MR5: *sin*(x) = -*sin*(2 π - x)

…

Priorization Approaches

- Usage profile
- Algorithm

Usage Profile

- *Restricted* use of programs – interested in a subset of properties

Usage Profile

- *sin*(x)
  - Electrical Engineers
    - *sin*(x) = *sin*(x + 2 π)
  - Land Surveyors
    - *sin*(-x)= -*sin*(x)
    - *sin*(x) = *sin*(π-x)

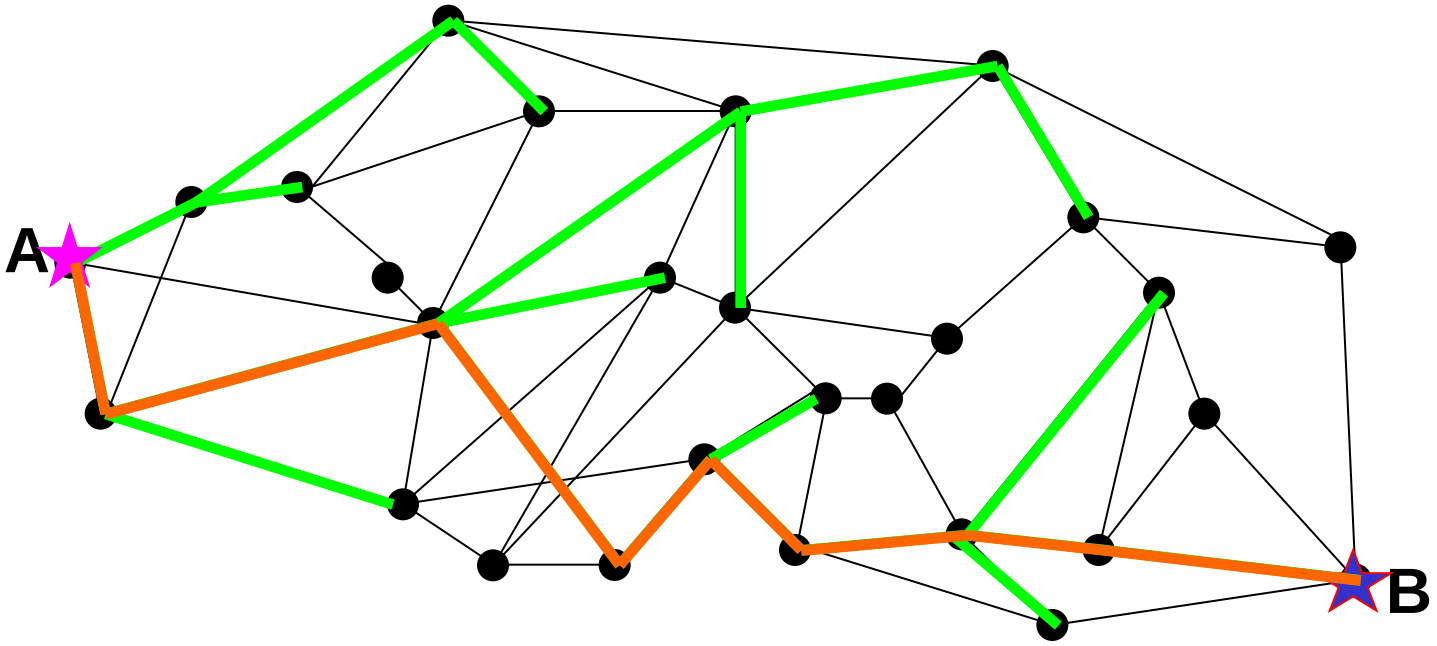
Algorithm

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

Example

- Shortest Path problem:

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



Example

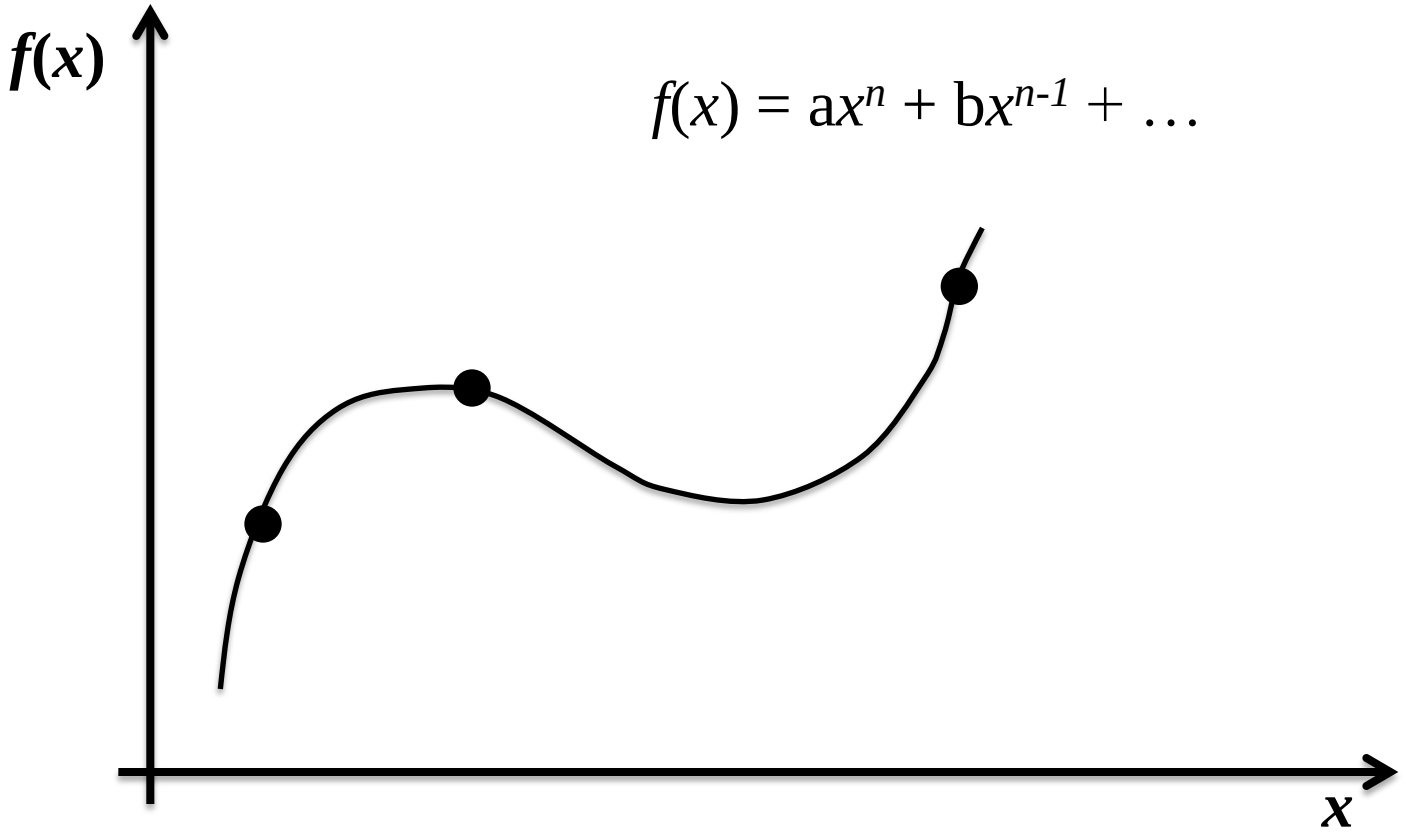
- $|SP(G, a, b)| = |SP(G', a', b')|$
- $|SP(G, a, b)| = |SP(G, a, c)| + |SP(G, c, b)|$
- $|SP(G, a, b)| = |SP(G, b, a)|$

Fault-Detection Effectiveness

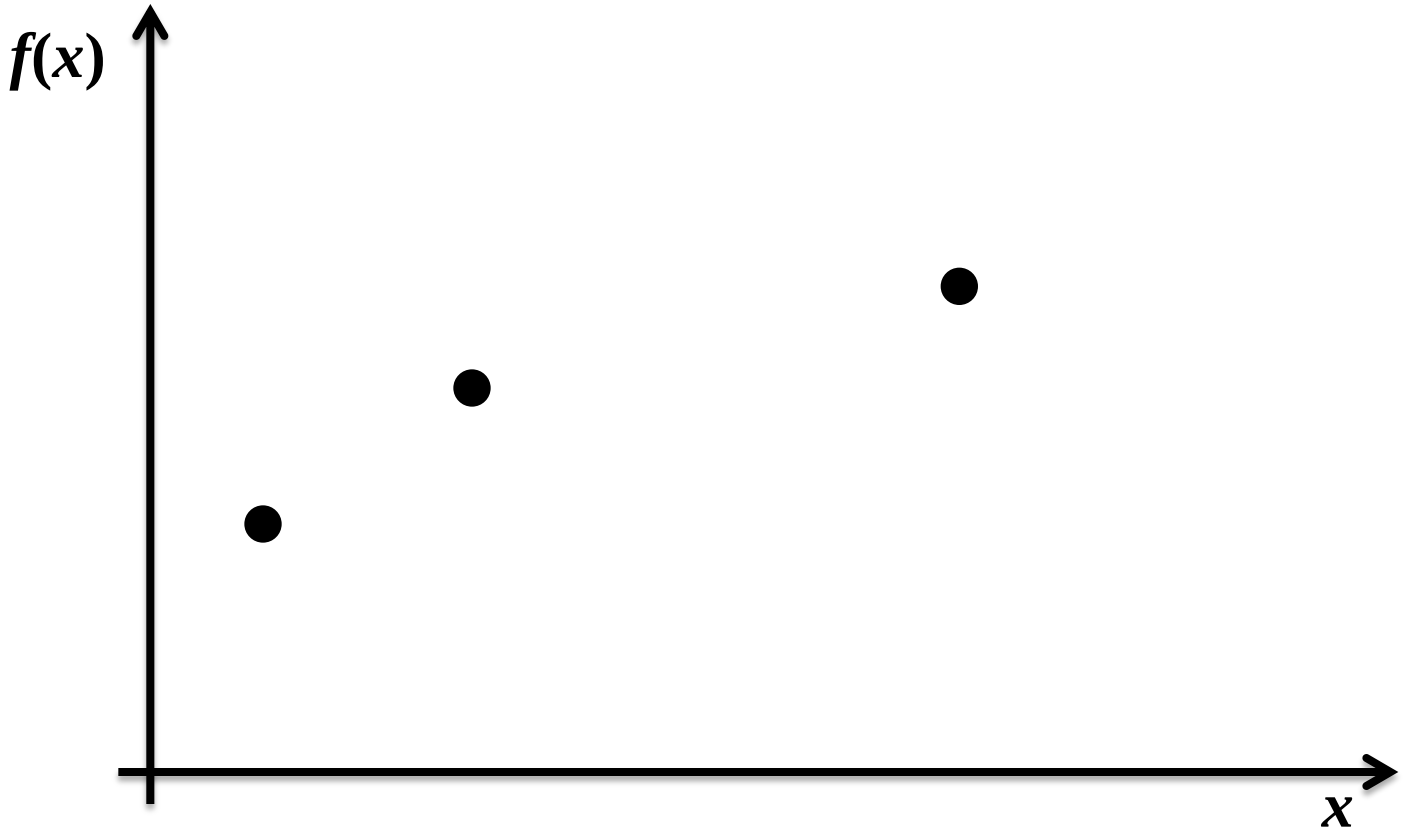
How many MRs are required?

Empirical Observation:

a few diverse MRs



$f(x) = ax^n + bx^{n-1} + \ldots$


f(x)
x

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

End-User Software Engineering

- Limited knowledge of testing
- Unaccessibility to testing tools
- Need a testing method
  - easy to learn
  - easy to use
  - easy to automate

End-User Software Engineering

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

Diversity

the key underlying concept in test case selection strategies

Diversity

- Success of MT in revealing faults undetected by other conventional testing methods
- Diverse MRs in MT

Diversity

underlying concept in software testing

Conclusion

Simplicity

Thanks!

References:

- Metamorphic Testing: A Literature Review, S. Segura, A. B. Sanchez and A. Ruiz-Cortes, Technical Report ISA-15-TR-01, University of Seville, 2015.