Lecture 4

Metamorphic Testing - II

1

Metamorphic Testing

- A property-based method to generate test cases
- A simple but effective method to alleviate the test oracle problem

2

- Testing can demonstrate the presence of faults but not the absence of faults
- Are successful test cases really useless?

3

Intuition

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 1

Given a program P to find the sum of a series of numbers (L)

Suppose P(L) outputs a sum of 123,456

Is 123, 456 correct?

5

Example 1 (continued)

The summation of a series of numbers has a property - commutative

Suppose P(L) outputs a sum of 123,456

Is 123, 456 correct?

6

Example 1 (continued)

Let L' be a permutation of L.

Then, we expect P(L) and P(L') return the same output if P is correctly implemented

What is the implications of

- P(L) = P(L')
- P(L) =\= P(L')

7

Example 1 (continued)

What is the cost?

8

Example 2

Suppose *sin*(29.8) returns 0.49876

(assume the input to sin is in the unit of degree)

Is the answer correct?

9

Example 2 (continued)

- *sin* function has the following properties
  - *sin*(x)= *sin*(x+360)

10

Example 2 (continued)

- *sin* function has the following properties
  - *sin*(x)= *sin*(x+360)
- Compute 29.8 + 360 = 389.8
- Execute the program with 389.8 as input
- Check whether *sin*(29.8) = *sin*(389.8)

11

Example 3

- Shortest path program *SP*(*G*, *a*, *b*) which returns a path from node *a* to node *b* in graph *G*.
- Suppose the program returns:
  - |*SP*(*G*, *a*, *b*)| = 1,234,567 correct or incorrect?

    where |*SP*(*G*, *a*, *b*)| denotes the length of *SP*(*G*, *a*, *b*)

12

Example 3 (continued)

- Shortest Path Problem has the following properties:
  - $|SP(G, a, b)| = |SP(G, b, a)|$
  - $|SP(G, a, b)| = |SP(G, a, c)| + |SP(G, c, b)|$ where c is a node in $SP(G, a, b)$

13

Example 3 (continued)

- Consider the property:
  - $|SP(G, a, b)| = |SP(G, b, a)|$

- Execute SP with input (G, b, a)
- Check whether $|SP(G, a, b)| =?= |SP(G, b, a)|$

14

Example 3 (continued)

- Consider the property
  - $|SP(G, a, b)| = |SP(G, a, c)| + |SP(G, c, b)|$ where c is a node in $SP(G, a, b)$
- Execute SP with inputs (G, a, c) and (G, c, b)
- Check whether

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

15

Example 3 (continued)

Can we use the following properties?

- $SP(G, a, b) = SP(G, b, a)|$
- $SP(G, a, b) = SP(G, a, c) + SP(G, c, b)$ where c is a node in $SP(G, a, b)$

16

Process of Metamorphic Testing

17

Example 2 (continued)

- An existing test case of 29.8; execute the program with it
- A property
  - *sin*(x)= *sin*(y) if y= x+360
- For x= 29.8, compute y=29.8+360 = 389.8
- Execute the program with 389.8 as input
- Check whether *sin*(29.8) = *sin*(389.8)

18

Metamorphic Testing (A Simplified Form)

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

19

Example 2 (continued)

- Suppose *sin*(29.8) returns 0.49876
  29.8 is the source test case
- *sin* function has the following properties
  - [***If y*= x+360 then *sin*(x)= *sin*(y)] is the MR**
- Compute 29.8+ 360 = 389.8
  389.8 is the follow-up test case

20

Example 3 (continued)

- Consider the property:
  - $|SP(G, a, b)| = |SP(G, b, a)|$

- Execute *SP* with input (G, b, a)
- Check whether $|SP(G, a, b)| =?= |SP(G, b, a)|$

21

Example 3 (continued)

- Source test case is *(Graph1, node_3, node_62);*
  Execute *SP* with input *(Graph1, node_3, node_62)*
- Consider the property:
  - $|SP(G, a, b)| = |SP(G, b, a)|$
- Follow-up test case is *(Graph1, node_62, node_3)*
- Execute *SP* with input *(Graph1, node_62, node_3)*
- Check whether *(Graph1, node_3, node_62) =?=*
  (Graph1, node_62, node_3)

22

Example 3 (continued)

- Consider the property
  - $|SP(G, a, b)| = |SP(G, a, c)| + |SP(G, c, b)|$ where c is a node in $SP(G, a, b)$
- Execute *SP* with inputs (G, a, c) and (G, c, b)
- Check whether

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

23

Example 3 (continued)

- Source test case is *(Graph1, node_3, node_62);*

  Execute *SP* with *(Graph1, node_3, node_62);*

  The returned path is:

  node_3 –> node_6 -> ....node_97 -> ...node_62
- Consider the property:
  - $|SP(G, a, b)| = |SP(G, a, c)| + |SP(G, c, b)|$ where c is a node in $SP(G, a, b)$

24

Example 3 (continued)

- Follow-up test case are
  (Graph1, node_3, node_97) and
  (Graph1, node_97, node_62)
- Execute *SP* with inputs of
  (Graph1, node_3, node_97) and
  (Graph1, node_97, node_62)

25

Example 3 (continued)

- Check whether
  |(Graph1, node_3, node_62)|
  =?= *|(Graph1, node_3, node_97)|*
  + *|(Graph1, node_97, node_62)|*

26

Applications of MT

Cases of Successful Applications of MT

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

Interesting Results

Reveal undetected faults

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

Testing Compilers with MT

- 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 with MT

Their testing method is basically 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

Testing Compilers with MT

Reported to reveal over 100 faults in two popular C compilers:

GCC and LLVM

32

Metamorphic Relations (MRs)

33

Metamorphic Relations (MRs)

For a given algorithm, there are normally many MRs

MR1, MR2, MR3, ……..MRn

- sin(x) = sin(x+360)
- sin(-x) = - sin(x)
- sin(x) = sin(x+180)
- …….
- …….

34

Metamorphic Relations (MRs)

MR1, MR2, MR3, ……..MRn

- Which one to be used first?
- Which one more effective to reveal faults?

35

Identifications of MRs

36

Summary

37

Reference

- F. T. Chan, T. Y. Chen, S. C. Cheung, M. F. Lau and S. M. Yiu, Application of Metamorphic Testing in Numerical Analysis, *Proceedings of the IASTED International Conference on Software Engineering*, 191-197, 1998.
- S. Segura, G. Fraser, A. B. Sanchez and A. Ruiz-Cortes, A Survey on Metamorphic Testing, IEEE Transactions on Software Engineering, Vol. 42(9), 805-924, 2016.

38