Lecture

Mutation Testing - I

1

Mutation Testing

- To measure the effectiveness
- To measure the adequacy
- To generate test cases

2

Outline

- Mutants
- Mutation operators
- Killing mutants

3

Mutants

A slightly mutated version of a given program (or reference program)

4

Examples of Mutants

Given program	Mutant 1	Mutant 2
Input A, B	Input A, B	Input B, A
C= A + B	C= A * B	C= A + B
Output C	Output C	Output C

5

Examples of Mutants (continued)

Given program	Mutant 3	Mutant 4
Input A, B	Input A, B	Input A, B
C= A + B	C= 2 * B	C= A + B
Output C	Output C	Output B

6

Examples of Mutants (continued)

Given program	Mutant 5
……	……
If (A >= B)	If (A < B)
Then ….	Then …..
Else …	Else …..
…….	……
…….	…….

7

Examples of Mutants (continued)

Given program	Mutant 6
……	……
If (A >= B)	If (A >= B)
Then ….	Then …..
Else …	
C=A +B	C=A+B
…….	…….

8

Mutants

A slightly mutated version of a given program

9

Examples of Mutants (continued)

Given program

```
Integer A, B, C
String S
…..
C= A + B
……..
……..
```

```
Integer A, B, C
String S
……..
C= A + S
…….
…….
```

10

Mutants

A slightly mutated version of a given program (reference program)

- Successfully compiled

11

Mutants

A slightly mutated version of a given program (reference program)

- Successfully compiled

Through some transformation rules

  – Mutation operators

12

Mutation Operators

- Defining transformation rules to generate mutants

13

Mutation Operators (continued)

Examples

- Change of arithmetic operators

  A+B becoming A*B

  (arithmetic operators: +, -, *, /, **, …)

- Chang of arithmetic variables

  C=A+B becoming D=A+B

14

Mutation Operators (continued)

Examples

- Replacement of variables by constants

  A=A+B becoming A=A+1

15

Mutation Operators (continued)

Examples

- Change of relational operators

  (A+B >= A*B) becoming (A+B > A*B)

  (A+B ==A*B) becoming (A+B != A*B)

Arithmetic relational operators:

==, !=, >, <, >=, <=

16

Mutation Operators (continued)

Examples

- Change of logical operators

  (A+B >= A*B) && (A-B > A/B)

  becoming

  (A+B >= A*B) || (A-B > A/B)

where && means AND, || means OR

17

Mutation Operators (continued)

Examples

- Change of logical operators

  (A+B >= A*B) && (A-B > A/B)

  becoming

  !(A+B >= A*B) && (A-B > A/B)

where ! means negation

18

Mutation Operators (continued)

Examples

- Change of logical variables

  X && Y becoming X && Z

- Replacement of logical variables by Boolean constants

  X &&Y becoming X && true

19

Mutation Operators (continued)

Examples

While (A+B > C)

Do { ……...; …….; ;…..}

becoming

Repeat {…….; …..; ; ……}

Till (A+B > C)

20

Mutation Operators

- Rules to generate mutants
- Syntactic changes
- Programming language dependent

21

List of Mutation Operators

- Arithmetic operator replacement
  - %; **; ……
- Relational operator replacement
- Conditional operator replacement
- Assignment operator replacement
  - A += 5 (A = A + 5)
  - -=; *=; /=,

22

List of Mutation Operators (continued)

- Unary operator insertion
  - A= B * C becoming A= -B* C
  - While (A+B > C) becoming While !(A+B > C)
- Unary operator deletion
- Scalar variable replacement
  - Replaced by variable of the appropriate type

23

List of Mutation Operators (continued)

- Scalar variable replacement
  - Replaced by variable of the appropriate type
- Constant replacement
  - A=B*2 becoming A=B*3
- Array name replacement
  - Lists, matrices, …..

24

List of Mutation Operators (continued)

- Array reference for array reference replacement
  - A = B + C[3] becoming A = B + C [2]
- Exchange of constant and scalar variable
  - A = B * 2 and A = B * C
- Exchange of constant and array reference
  - A = B + C[3] and A = B + 3

25

List of Mutation Operators (continued)

- Exchange of scalar variable and array reference
  - A = B + C[3] and A = B + C
- Exchange of type declaration
  - Integer type declaration & string type declaration
- …….

26

Mutants

- A slightly mutated version of a given program through the application of mutation operators
- Successfully compiled
- Programming language dependent

27

Summary

28

References

- Y. Jia and M. Harman, “An Analysis and Survey of the Development of Mutation Testing”, IEEE Transactions on Software Engineering, Vol. 37(5), 649-678, 2011.

29