Tutorial 8

Swinburne University of Technology

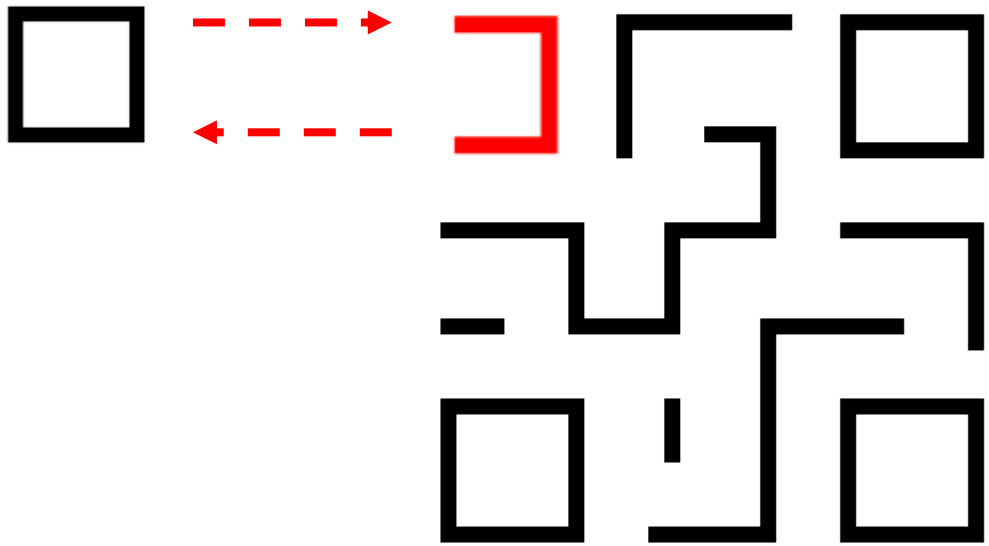
Software Testing and Reliability (SWE30009)

Semester 2, 2023

Lecturer: Prof T. Y. Chen

Tutor: Dr Hung Q Luu

Mutant



A mutant is a slightly modified version of a given program

Mutant

- Successfully compiled or interpreted mutants
  - Mutants that are non-trivial
- Equivalent mutants
  - Mutants that are equivalent to original program, e.g., A = B → A = B * 1
- Non-equivalent mutants
  - Mutants that are different from original program, e.g., A = B → A = B * 2

Some mutation operators

- Change of arithmetic operators
  - **A = B + C → A = B * C**
  - **A += 1 → A -= 1**
- Change of arithmetic variables
  - **A = B + C → A = B + D**
- Change a variable to a constant
  - **A = B + C → A = B + 1**
- Change a constant value
  - **A = B + 1 → A = B + 5**

Some mutation operators (continue)

- Change of relational operators (<,<=,==,!=,>=,>)
  - **D > E → D <= E**
- Change of logical operators (AND, OR, NEGATION)
  - **(A > 1) AND (2 > 1) → (A > 1) OR (2 > 1)**
- Change of logical statement
  - **(A > 1) AND (2 > 1) → (A > 1) AND False**
- Change of array index
  - **A = B[1] → A = B[2]**
- …

Mutation operators

- Mutation operators are systematic rules used to generate mutants
- There are different types of mutatation operators
- Each programming language has certain types of mutation operators

Killing mutants in conventional testing

- A mutant is said to be **killed** if its output differs the output of original program given the same test case.

Mutation score

- Assume we have
  - A set of **m** mutants, SM = {M1, M2, ...Mm}
  - A test suite of **n** test cases, TS = {TC1, TC2, ..., TCn}
- Mutant scores

$$MS = k / m$$

  - **k** – the number of killed mutants
  - **m** - the total number of (non-equivalent) mutants

Mutation score

- Depends on the set of mutants used
- Depends on the test suite used

Pratice

Test suite TS1
TC1: B = 2, C = 2
TC3: B = 3, C = 3

Test suite TS2
TC3: B = 5, C = 1
TC4: B = -1, C = 0

Program P
Input: B, C
A = B + C
Output: A

Mutant M1
Input: B, C
A = B - C
Output: A

Mutant M2
Input: B, C
A = B * C
Output: A

Mutant M3
Input: B, C
A = B + 3
Output: A

Mutant M4
Input: B, C
A = B + B
Output: A

Pratice

Test suite TS1
TC1: B = 2, C = 2
TC3: B = 3, C = 3

Test suite TS2
TC3: B = 5, C = 1
TC4: B = -1, C = 0

Program P
Input: B, C
A = B + C
Output: A

Mutant M1
Input: B, C
A = B - C
Output: A

Mutant M2
Input: B, C
A = B * C
Output: A

Mutant M3
Input: B, C
A = B + 3
Output: A

Mutant M4
Input: B, C
A = B + B
Output: A

TC1: B = 2, C = 2
P: A = 4
M1: A = 0 -> killed
M2: A = 4
M3: A = 5 -> killed
M4: A = 4

TC2: B = 3, C = 3
P: A = 6
M1: A = 0 -> killed
M2: A = 9 -> killed
M3: A = 6
M4: A = 6

Test suite TS1
M1: killed
M2: killed
M3: killed
M4: no killed
Mutation Score: 3/4 = 75%

TC3: B = 5, C = 1
P: A = 6
M1: A = 4 -> killed
M2: A = 5 -> killed
M3: A = 8 -> killed
M4: A = 10 -> killed

TC4: B = -1, C = 1
P: A = -1
M1: A = -1
M2: A = 0 -> killed
M3: A = 2 -> killed
M4: A = -2 -> killed

Test suite TS2
M1: killed
M2: killed
M3: killed
M4: killed
Mutation Score: 4/4 = 100%

System under test

p.py

```
10 # function to compute sum of a list
11 # note: for demonstration only and it is not optimal
12 def compute_sum(input, verbose=True):
13     result = input[0]
14     for id,n in enumerate(input):
15         if id>0:
16             #if isfloat(n): # optional
17                 result += n
18     if verbose:
19         print('Input: {}\nOutput: {}'.format(input,result))
20     return result
21 
22 
23 # main program
24 if __name__ == "__main__":
25     import argparse
26     params = argparse.ArgumentParser()
27     params.add_argument('-input', nargs="+", type=float)
28     args = params.parse_args()
29     input = args.input
30     output = compute_sum(input)
```

Mutants

- Original program P:

S = A1 + A2 + … + An

- Mutant M1:

S = A1 * A2 * … * An

- Mutant M2:

S = A2 + … + An

- Mutant M3:

S = 3 + A2 + … + An

Test cases

- Test case TC1

  Input: **[3, 5, 4]**

- Test case TC2

  Input: **[0, 10, 3]**

- Test case TC3

  Input: **[2, 2]**

- Test case TC4

  Input: **[1, 2, 3]**

Mutation testing with test oracle

SUT	P output	M1 output	M2 output	M3 output
Test cases	S = A1 + A2 + … + An	S = A1 * A2 * … * An	S = A2 + … + An	S = 3 + A2 + … + An
[3, 5, 4]				
[0, 10, 3]				
[2, 2]				
[1, 2, 3]				

Mutation testing with test oracle

SUT	P output	M1 output	M2 output	M3 output
Test cases	S = A1 + A2 + … + An	S = A1 * A2 * … * An	S = A2 + … + An	S = 3 + A2 + … + An
[3, 5, 4]	12	60	**9**	12
[0, 10, 3]	13	0	**13**	**16**
[2, 2]	4	4	2	**7**
[1, 2, 3]	6	6	**5**	**8**

Mutation testing with test oracle

SUT	P output	M1 output	M2 output	M3 output
Test cases	S = A1 + A2 + … + An	S = A1 * A2 * … * An	S = A2 + … + An	S = 3 + A2 + … + An
[3, 5, 4]	12	killed	killed	-
[0, 10, 3]	13	killed	-	killed
[2, 2]	4	-	killed	killed
[1, 2, 3]	6	-	killed	killed

Discussion with conventional testing

- What is the mutation score for a test case?
- What is the averaged mutation score for all test cases?
- What is the mutation score for test suite combining all test cases?

Discussion with conventional testing

- What is the mutation score for a test case?

$$\mathbf{MS = k / m}$$

k is number of killed mutants by the test case (on the set of m mutants)

- What is the averaged mutation score for all test cases?

$$\mathbf{MS_{average} = (k_1 + k_2 + \ldots + k_n) / (m * n)}$$

k_i is number of killed mutants by a test case i (on the set of m mutants x n test cases)

- What is the mutation score for test suite combining all test cases?

$$\mathbf{MS_{test\_suite} = (\delta_1 + \delta_2 + \ldots + \delta_m) / m}$$

δ_j = 1 if mutant j is killed by any test case, and 0 otherwise (on the set of m mutants x n test cases)

Killing mutants in metamorphic testing

- A mutant is said to be **killed** if the relation of a test group and its outputs violates the MR.

Mutation testing with MT

MR1: Adding a new number to the sum

sum ([A1, A2, …, An]) + B = sum ([A1, A2, …, An, B])

		SUT	M1 output	M2 output	M3 output
Source test case	Follow-up test case	Relation	S = A1 * A2 * … * An	S = A2 + … + An	S = 3 + A2 + … + An
[3, 5, 4]	[3, 5, 4, 1]	sum([3,4,5]) + 1 = sum([3,4,5,1])			
[0, 10, 3]	[0, 10, 3, 0]	sum([0,10,3]) + 0 = sum([0,10,3,0])			
[2, 2]	[2, 2, 3]	sum([2,2]) + 0 = sum([2,2,3])			
[1, 2, 3]	[1, 2, 3, 4]	sum[1,2,3]) + 4 = sum[1,2,3,4])			

Mutation testing with MT

MR1: Adding a new number to the sum

sum ([A1, A2, …, An]) + B = sum ([A1, A2, …, An, B])

		SUT	M1 output	M2 output	M3 output
Source test case	Follow-up test case	Relation	S = A1 * A2 * … * An	S = A2 + … + An	S = 3 + A2 + … + An
[3, 5, 4]	[3, 5, 4, 1]	sum([3,4,5]) + 1 = sum([3,4,5,1])	61 =/= 60	**10=10**	13 == 13
[0, 10, 3]	[0, 10, 3, 0]	sum([0,10,3]) + 0 = sum([0,10,3,0])	0 == 0	**13=13**	16 == 16
[2, 2]	[2, 2, 3]	sum([2,2]) + 0 = sum([2,2,3])	7 =/= 12	5 == 5	8 == 8
[1, 2, 3]	[1, 2, 3, 4]	sum[1,2,3]) + 4 = sum[1,2,3,4])	10 =/= 24	**10=/=24**	12 == 12

Mutation testing with MT

MR1: Adding a new number to the sum

sum ([A1, A2, …, An]) + B = sum ([A1, A2, …, An, B])

		SUT	M1 output	M2 output	M3 output
Source test case	Follow-up test case	Relation	S = A1 * A2 * … * An	S = A2 + … + An	S = 3 + A2 + … + An
[3, 5, 4]	[3, 5, 4, 1]	sum([3,4,5]) + 1 = sum([3,4,5,1])	killed	-	-
[0, 10, 3]	[0, 10, 3, 0]	sum([0,10,3]) + 0 = sum([0,10,3,0])	-	-	-
[2, 2]	[2, 2, 3]	sum([2,2]) + 0 = sum([2,2,3])	killed	-	-
[1, 2, 3]	[1, 2, 3, 4]	sum[1,2,3]) + 4 = sum[1,2,3,4])	killed	**killed**	-

Discussion with MT

- What is the mutation score for a test group (consists of two/more test cases)?
- What is the averaged mutation score for all test groups?
- What is the mutation score for test suite combining all test groups?

Discussion with MT

- What is the mutation score for a test group (consists of two/more test cases)?

$$\mathbf{MS = k / m}$$

k is number of killed mutants by the test group (on the set of m mutants)

- What is the averaged mutation score for all test groups?

$$\mathbf{MS_{average} = (k_1 + k_2 + \ldots + k_n) / (m * n)}$$

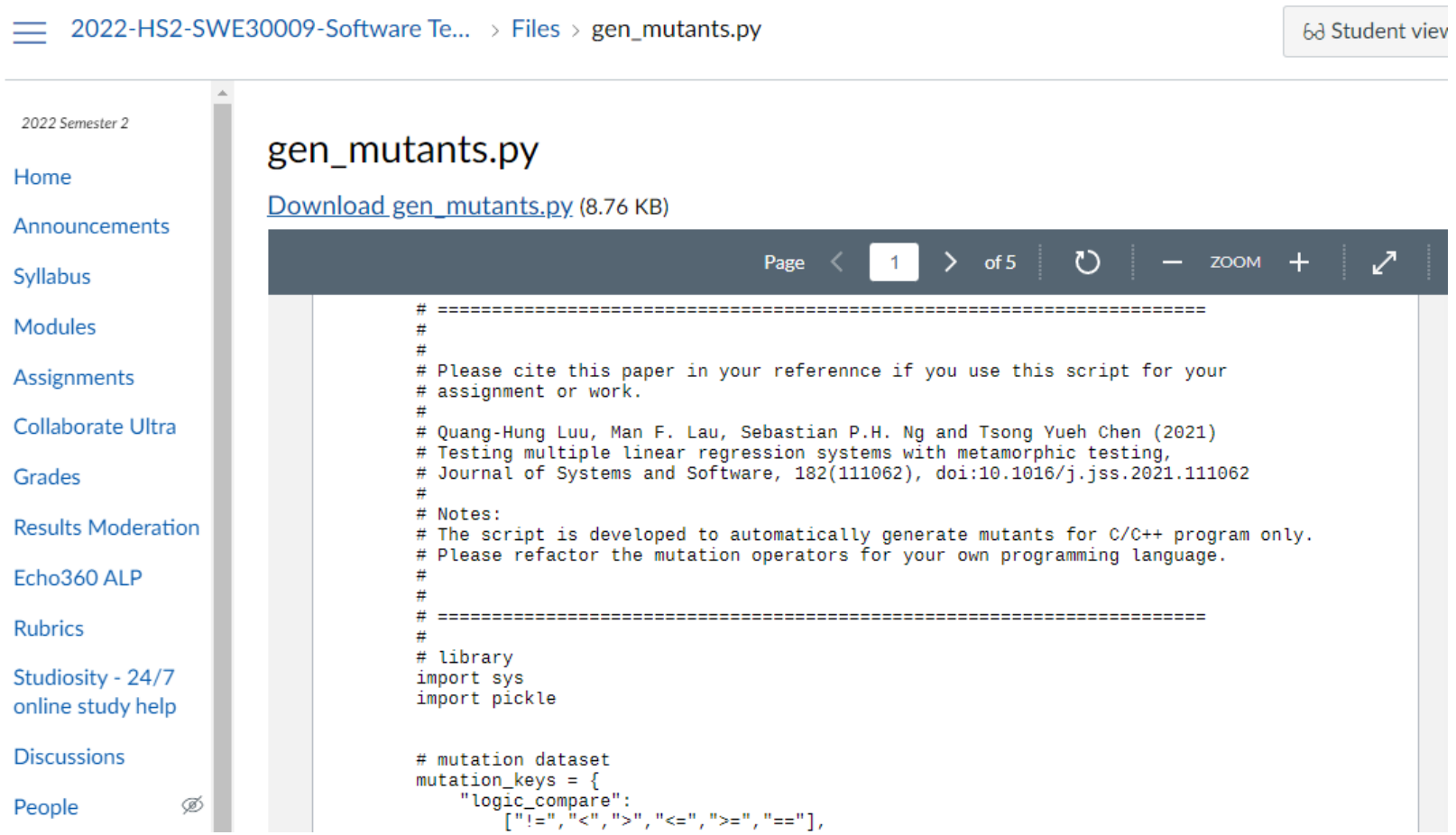
k_i is number of killed mutants by a test group i (on the set of m mutants x n test groups)

- What is the mutation score for test suite combining all test groups?

$$\mathbf{MS_{test\_suite} = (\delta_1 + \delta_2 + \ldots + \delta_m) / m}$$

δ_j = 1 if mutant j is killed by any test group, and 0 otherwise (on the set of m mutants x n test groups)

Mutation tool



Mutation tool

```
"logic_bool":
    ["true","false"],
"logic_return":
    ["?","&&false?","||true?"],
"math_operator":
    ["+","-","*","/","%"],
"math_increment":
    ["++","--","+=2","-=2"],
"math_value":
    ["=0","=1","=2"],
"math_numeric":
    ["+1","-1","+2","-2"],
"condition_if":
    ["if(","if(!","if(true||","if(false&&"],
"condition_while":
    ["while(","while(!","while(~","while(false&&"],
"condition_loop":
    ["break;","continue;","{;}"],
```

```
if __name__ == "__main__":
    if len(sys.argv)==2:
        main(sys.argv[1],"./",0,-1)
    if len(sys.argv)==3:
        main(sys.argv[1],sys.argv[2])
    if len(sys.argv)==5:
        main(sys.argv[1],sys.argv[2],int(sys.argv[3]),int(sys.argv[4]))
    else:
        print("Usage: python gen_mutatation.py <file-to-mutate.c> [folder-to-store-
mutated-files] [start-line] [end-line]")
        print("Example: python gen_mutation.py ../cpp/mlr.c mutations/svd/ 10 440")
```