ISSN:2229-6093

Shalini Kapoor, Int. J. Comp. Tech. Appl., Vol 2 (5), 1206-1211

TEST CASE EFFECTIVENESS OF HIGHER ORDER MUTATION TESTING

Shalini Kapoor
CSE Deptt, GNI, Mullana
get_shalini@rediffmail.com

***Abstract*--- Effectiveness means how good a test case is in finding faults. Traditional mutation testing considers First Order Mutants (FOM) created by injection of a single fault. We focus on Higher Order Mutants (HOM) and in particular on subsuming HOM. Higher Order Mutants contain more than one fault. We report in this paper that a strongly subsuming HOM is more effective as it kills all the FOM's from which it is constructed thereby reducing testing efforts without loss of effectiveness.**

Keywords—Mutation Testing; First order mutants; Higher order mutants

I. INTRODUCTION

Mutants can be classified into two types: First Order Mutants (FOMs) and Higher Order Mutants (HOMs). FOMs are generated by applying mutation operators only once. HOMs are generated by applying mutation operators more than once. This paper introduces the concept of subsuming HOMs. A subsuming HOMs is harder to kill than the FOMs from which it is constructed. As such, it may be preferable to replace the FOMs with the single HOM. In particular, the paper introduces the concept of a strongly subsuming HOMs. A subsuming HOMs is only killed by a subset of the intersection of test cases that kill each FOM from which it is constructed.

Consider a subsuming, h, constructed from the FOMs f1... fn. The set of test cases that kill h also kill each and every FOM f1... fn. Therefore, h can replace all of the mutants f1... fn without loss of test effectiveness. The converse does not hold; there exist test sets that kill all FOMs f1... fn but which fail to kill h. The FOMs cannot, even taken collectively, replace the HOM without possible loss of test effort. This is the sense in which h can be said to 'strongly subsume' f1... fn.

II. CLASSIFICATION OF HIGHER ORDER MUTANTS

HOMs can be classified in terms of the way that they are 'coupled' and 'subsuming', as shown in Figure 1. In Figure 1, the region area in the central Venn diagram represents the domain of all HOMs. The sub-diagrams surrounding the central region illustrate each category. For sake of simplicity of exposition these examples illustrate the second order mutant case; one that assumes that there are two FOMs f1 and f2, and h denotes the HOM constructed from the FOMs f1 and f2. The two regions depicted by each sub diagram represent the test sets containing all the test cases that kill FOMs f1 and f2. The shaded area represents the test set that contains all test cases that kill HOM h. The areas of the regions indicate the proportion of the domain of HOMs for each category. Following the coupling effect hypothesis, if a test set that kills the FOMs also contains cases that kill the HOM, we shall say that the HOM is a 'coupled HOM', otherwise we shall say it is a 'de-coupled HOM'. Therefore, in Figure 1, the sub-diagram is a coupled HOM if it contains an area where the shaded region overlaps with the unshaded regions. For example the sub-diagrams (a), (b) and (f). Since the shaded region from sub-diagrams (c) and (d) do not overlap with the unshaded regions, (c) and (d) are de-coupled HOMs. Subdiagram (e) is a special case of a de-coupled HOM, because there is no test case that can kill the HOM; there is no overlap, the HOM is an equivalent mutant.

Subsuming HOMs, by definition, is harder to kill than their constituentFOMs. Therefore, in Figure 1, the subsuming HOMs can be represented as those where the shaded area is smaller than the area of the union of the two unshaded regions, such as sub-diagrams (a), (b) and (c). By contrast, (d), (e) and (f) are non-subsuming. Furthermore, the subsuming HOMs can be classified into strongly subsuming HOMs and weakly subsuming HOMs. By definition, if a test case kills a strongly subsuming HOM, it guarantees that its constituent FOMs are killed as well. Therefore, if the shaded region lies only inside

the intersection of the two unshaded regions, it is a strongly subsuming HOM, depicted in (a), otherwise, it is a weakly subsuming HOM, depicted in (b) &(c).

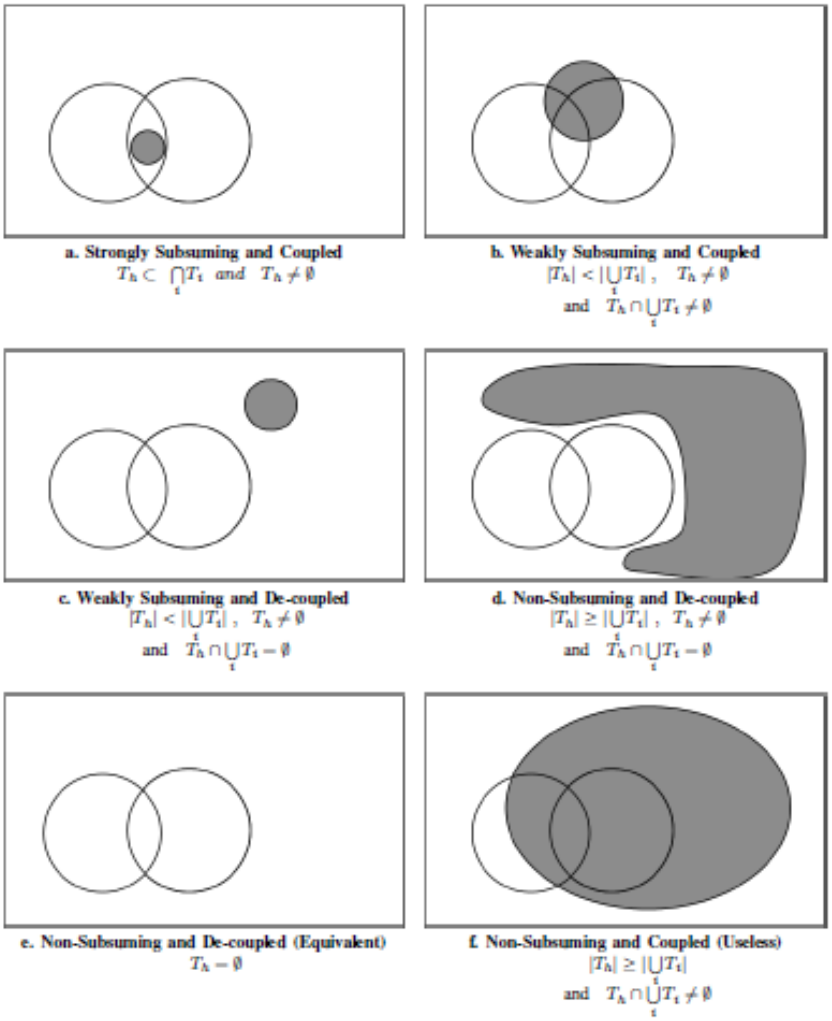


Fig. 1. HOMs Classification. These pseudo Venn diagrams depict the relationship between some of the interesting classes of HOMs and their first order constituents. In each of the six examples, the rectangle depicts all possible test inputs. The two circles within each rectangle depict the possible regions of test cases that can kill a FOM in the standard way for a Venn Diagram. The shaded region indicates the killing test sets for the HOMs. The size of the shaded region is intended to show the relative size of the HOM kill set compared to the FOM kill sets. Because it attempts to show set size, the diagram is a "pseudo" Venn Diagram, rather than a true Venn Diagram. For ease of exposition, the diagrams illustrate only the second order case, whereas the definitions cover arbitrary order. HOMs of type (a), (b) and (c) are harder to kill than their constituent FOMs, thereby capturing potentially subtler faults. In particular, type (a) are both subtle and useful; they can replace their constituent FOMs because they are killed by a subset of the intersection of test cases that kill their constituents. For a HOM h, constructed from FOMs $f_1,...,f_n$, the test set T_h contains all the test cases that kill h, while the test sets $T_1,...,T_n$ are the test sets that kill that kill $f_1,...,f_n$ respectively.

III. REASONS TO SUPPORT HOM

1) Cost: Work on Mutant Sampling and Selective Mutation has shown how the number of mutants can be reduced with only a small impact on test effectiveness [1], [8], [7], [17].

2) Uncertainty: Work on reducing the impact of equivalent mutants has reduced, though not eradicated, this problem [6], [16], [5], [4], [1].

3) Realism: Empirical evidence has been provided that the faults denoted by mutants do, indeed, overlap with a class of real faults [16], [12], [5].

IV. PROPOSED WORK

In order to explain Higher Order Mutation Testing we take the following example.

Original_Program

{

if ((a>b) && (a>c))

……

}

We create a first order mutant of the Original_Program by adding a single fault i.e. we change a>b to a<b and name it FOM1 as below:

FOM1

{

if ((a<b) && (a>c))

……

}

We create another first order mutant of the Original_Program by adding a single fault i.e. we change a>c to a<c and name it FOM2 as below:

FOM2

{

if ((a>b) && (a<c))

……

}

FOM1 and FOM2 are first order mutants as they vary by a single fault from the Original_program. Now we create a HOM from FOM1 and FOM2 by having more than one fault from the Original_program .Our HOM differs from original program by 2 faults. We change a>b to a<b and also a>c to a<c.

HOM

{

if ((a<b) && (a<c))

……

}

Now we prove that if we are able to find a subsuming HOM in particular a strongly subsuming HOM, it will kill all the FOM's from which it is constructed thereby reducing the number of test cases without loss of test case effectiveness.

We take simple example to find largest of 3 numbers a, b and c. Our program takes as input 3 numbers and as output it gives the largest of these numbers.

Original_program

#include<stdio.h>

#include<conio.h>

void main ()

{

int a,b,c;

clrscr();

printf("ENTER THE 3 NUMBERS");

scanf("%d%d%d",&a,&b,&c);

if((a>b) &&(a>c))

printf("A IS GREATEST");

else if ((b>a) &&(b>c))

printf("B IS GREATEST");

else if ((c>a) && (c>b))

printf("C IS GREATEST");

else printf (" WRONG RESULT");

}

We create First Order Mutant FOM1 from the Original_program by changing a> b to a<b

FOM1

```
if((a<b) &&(a>c))
printf("A IS GREATEST");
else if ((b>a) &&(b>c))
printf("B IS GREATEST");
else if ((c>a) && (c>b))
printf("C IS GREATEST");
else printf(" WRONG RESULT");
```

We create another First Order Mutant FOM2

from the Original_program by changing a> c to a<c

FOM2

```
if((a>b) &&(a<c))
printf("A IS GREATEST");
else if ((b>a) &&(b>c))
printf("B IS GREATEST");
else if ((c>a) && (c>b))
printf("C IS GREATEST");
else printf(" WRONG RESULT");
```

We create Higher order mutant HOM from First order mutant FOM1 and FOM2 by changing a> b to a<b and a>c to a<c. This differs from Original_program by 2 faults.

HOM

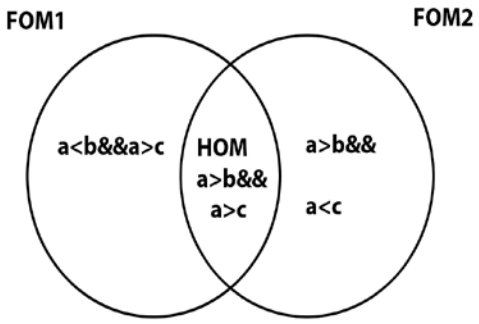
```
if((a<b) &&(a<c))
printf("A IS GREATEST");
else if ((b>a) &&(b>c))
printf("B IS GREATEST");
else if ((c>a) && (c>b))
printf("C IS GREATEST");
else printf(" WRONG RESULT");
```

Table 1

Test Cases which kill FOM1, FOM2 and HOM

Mutant	Test Case	Original_program RESULT	Mutant RESULT
FOM1	a>b && a>c	A IS GREATEST	WRONG RESULT
	a<b && a>c	WRONG RESULT	A IS GREATEST
FOM2	a>b && a>c	A IS GREATEST	WRONG RESULT
	a>b && a<c	WRONG RESULT	A IS GREATEST
HOM	a>b && a>c	A IS GREATEST	WRONG RESULT

Figure 2



From the Table-1 we find that there are 2 test cases which kill FOM1 and also 2 test cases which kill FOM2. There is one test case which kills HOM and is found in the intersection of FOM1 and FOM2. This test case kills both FOM1 and FOM2. The converse is not true. The test case ((a<b) && (a>c)) which kills FOM1 does not kill FOM2 and HOM. Similarly the test case ((a>b) && (a<c)) which kills FOM2 does not kill FOM1 and HOM. This is shown diagrammatically above in Figure 2.

Table 2

Test Data a,b,c	Original_program Result	FOM1 Result	FOM2 Result	HOM Result
15,10,5 a>b && a>c	A IS GREATEST	INVALID RESULT	INVALID RESULT	INVALID RESULT
25,20,10 a>b && a>c	A IS GREATEST	INVALID RESULT	INVALID RESULT	INVALID RESULT
20,25,10 a<b && a>c	INVALID RESULT	A IS GREATEST	INVALID RESULT	INVALID RESULT
25,20,50 a>b && a<c	INVALID RESULT	INVALID RESULT	A IS GREATEST	INVALID RESULT

From the Table-2 its clear that the test case a<b && a>c which kills FOM1 and the Original_program does not kill FOM2 and HOM, similarly the test case a>b && a<c which kills FOM2 and the Original_program does not kill FOM1 and HOM but the test case a>b &&a>c found in the intersection of FOM1 and FOM2 kills our HOM and also FOM1 and FOM2. So if use this test case automatically both FOM1 and FOM2 will get killed thereby reducing the number of test cases without leading to loss of effectiveness.

V. CONCLUSION

In this paper, we examine the utilities of Higher Order Mutation Testing by creating mutants both First and Higher Order. From the above work we conclude that though HOM's are harder to kill but if we are able to find a subsuming HOM than the number of test cases reduces as these test cases will automatically kill all the FOM's from which it is constructed thereby leading to reduction in test efforts without loss of effectiveness.

VI. REFERENCES

[1] A. T. Acree. On Mutation. Phd thesis, Georgia Institute of Technology, Atlanta, Georgia, 1980.
[2] T. A. Budd. Mutation Analysis of Program Test Data. Phd thesis, Yale University, New Haven, Connecticut, 1980.
[3] W. E.Wong. On Mutation and Data Flow. Phd thesis, Purdue University, West Lafayette, Indiana, 1993.
[4] A. P. Mathur and W. E. Wong. An Empirical Comparison of Mutation and Data Flow Based Test Adequacy Criteria. Technique report, Purdue University, West Lafayette, Indiana, 1993.
[5] A. S. Namin and J. H. Andrews. On Sufficiency of Mutants. In Proceedings of the 29th International Conference on Software Engineering (ICSE COMPANION'07), pages 73–74, Minneapolis, Minnesota, 20-26 May 2007.
[6] A. P. Mathur. Performance, Effectiveness, and Reliability Issues in Software Testing. In Proceedings of the 5th International Computer Software and Applications Conference (COMPSAC'79), pages 604–605, Tokyo, Japan, 11-13 September 1991.
[7] M. Sahinoglu and E. H. Spafford. A Bayes Sequential Statistical Procedure for Approving Software Products. In Proceedings of the IFIP Conference on Approving Software Products (ASP'90), pages 43–56 Garmis Partenkirchen, Germany, September 1990. Elsevier Science.
[8] R. A. DeMillo, D. S. Guindi, K. N. King, W. M. McCracken, and A. J. Offutt. An Extended Overview of the Mothra Software Testing Environment. In Proceedings of the 2nd Workshop on Software Testing, Verification, and Analysis (TVA'88), pages 142–151, Banff Alberta, Canada, July 1988. IEEE Computer society.
[9] A. J. Offutt, G. Rothermel, and C. Zapf. An Experimental Evaluation of Selective Mutation. In

Proceedings of the 15th International Conference on Software Engineering (ICSE'93), pages 100–107, Baltimore, Maryland, May 1993. IEEE Computer Society Press.

[10] W. E. Wong and A. P. Mathur. Reducing the Cost of Mutation Testing: An Empirical Study. Journal of Systems and Software, 31(3):185–196, December 1995.

[11] K. N. King and A. J. Offutt. A Fortran Language System for Mutation- Based Software Testing Software: Practice and Experience, 21(7):685–718, October 1991

[12] E. S. Mresa and L. Bottaci. Efficiency of Mutation Operators and Selective Mutation Strategies: An Empirical Study. Software Testing, Verification and Reliability, 9(4):205–232, December 1999.

[13] A. S. Namin and J. H. Andrews. Finding Sufficient Mutation Operators via Variable Reduction. In Proceedings of the 2nd Workshop on Mutation

Analysis (MUTATION'06), page 5, Raleigh, North Carolina, November

2006. IEEE Computer Society.

[14] A. S. Namin and J. H. Andrews. On Sufficiency of Mutants. In Proceedings of the 29th International Conference on Software Engineering (ICSE COMPANION'07), pages 73–74, Minneapolis, Minnesota, 20-26 May 2007.

[15] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf. An Experimental Determination of Sufficient Mutant Operators. ACM Transactions on Software Engineering and Methodology, 5(2):99–118, April 1996.

VII. AUTHORS PROFILE

.

Shalini Kapoor received the bachelor degree in Computer Science and Engineering from Haryana Engineering College, Jagadhri, India in 2003. She received her Master degree in Information Technology from Karnataka State University, Mysore, India in 2011. She has 2.5 years industrial experience and 4.5 years teaching experience. Presently she is working in Computer Science and Engineering Department of Guru Nanak Institutions Mullana