Orthogonality of Mutation Operators

Nipur*
Rakesh Kumar**
Priyanka Gupta***

Abstract

Mutation testing has been proved to be a good technique in revealing faults in different paradigms. But the computational expenses of using the technique are the major hurdle in its practical usage. Selective mutation has been proposed as an alternative to reduce the cost by many researchers. Orthogonality is a common programming language feature which exhibits linear independence in some sense. This paper suggests the use of Orthogonality of mutation operators, to select mutation operators with unique functionality which in turn can help to reduce the cost of mutation testing. Orthogonality of different mutation operators in procedural, object oriented and declarative paradigms has been discussed.

Keywords: Cost Reduction for Mutation Testing, Mutation Analysis, Orthogonal Mutation Operators.

Introduction

Mutation testing is a fault based technique that removes the pre specified faults introduced by programmers due to minute changes in language syntax. The research in this area started in late 1970's and still in process by number of researchers. Mutation has proved to be of great help in finding errors that number of test criteria has failed to observe. It makes use of mutation operators for that purpose. In order to go for mutation analysis of even a small program a lot of computation effort is required. The complexity of mutation is $O(n^2)$ where n is the number of variable references in program as found in Mathur and Wong(1993). Many possible changes can be introduced in a small sized program that gives rise to numerous mutants of the original program. Even for a small program that represents min function and has only 3 lines 44 mutants can be generated as found in Offutt et. al (1996). All these mutants have to be executed again and again against different test cases in order to make efficient, effective and complete test sets and for removal of errors. This enhances the running costs of usage of mutation testing and makes it computationally expensive. And that puts a question mark on practical usage of mutation testing. There is a need for proper solution to this problem. Any possible solution should reduce the cost without loosing on mutation adequacy criteria.

Nipur*
GKV, Dehradun

Rakesh Kumar**
KUK University, Kurukshetra

Priyanka Gupta***
MAIMT, Jagadhri

Selective Mutation

Mathur and Wong (1993) has proposed randomly selected x% mutation and constrained mutation criteria to reduce the number of mutants to be examined and number of test cases to be generated for the reduction in cost of mutation testing. Selective mutation has been suggested as an option in this direction by Offutt et al. (1996) and Offutt et al. (1993). Selective mutation concentrates on reducing the number of mutants generated by selecting some mutant operators and penalizing the others. Ofutt et al.(1996) has done experimental evaluation of Mothra's 22 mutation operators and concluded that 5 of these suffice to effectively implement mutation testing. It has been shown that in terms of data references standard mutation takes quadratic time and selective mutation can be effectively implemented in linear time . Also it has been shown in experimental results that on average 50% saving in execution efforts can be done with the help of selective mutation by Offutt et al.(1993). This is a major step towards the practical application of mutation testing.

Orthogonality and Orthogonal Mutation Operators

Orthogonality of a language means independence of features and properties. Also there should not be too many ways of doing the same thing (Gupta(2004)). Experimental results with Selective Mutation testing have assured this thing that subset of mutation operators can perform equally efficient unit testing. This indicates that mutation operators are not always orthogonal .Operator M1 is said to be orthogonal to operator M2 if test cases that kill mutants generated with operator M1 cannot kill mutants generated with operator M2, and vice versa as found in Lee et al.(2004). Therefore orthogonality relates to performance as well as to the selectiveness of mutation operators. To make practical application of mutation testing, orthogonality of mutation operators can be a good measure. If all operators placed in the set are orthogonal then it means that no more operator can be deleted without loosing on the mutation criteria. Therefore the study is worth in order to reduce the cost of mutation testing.

Orthogonality of Procedural Mutation Operators

Mathur and Wong (1995) have worked on selective mutation testing and selected some different operators which were efficient as well as sufficient, that relates directly to orthogonality of operators. They have worked upon x% randomly selected mutation and constrained mutation. They have investigated the effects of varying the percentage of mutants selected from 10% to 100%. They have selected abs and ror operators for constrained mutation experimentation. They have used a set of four programs (find, strmat1,strmat2, textfmt). They have generated mutants using all mothra operators except the go to operator. They have given the following conclusions based on mutation scores.

1) using x% mutation adequacy test sets

- For Find, Strmat1, Strmat2: 1) none of the x% mutation adequate test set is mutation adequate. 2)10%, 15%, 20% Mutant adequate test sets give a mutation score of at least 96.14%, 95.21% & 94.84% respectively. 3) 25%, 30%, 35% & 40% mutation adequate test set provide more than 98% mutation coverage.
- For textfmt: 1) 2 out of 5 40% Mutation adequate test sets are mutation adequate. 2) all 40% Mutation adequate test sets provide more than 98.15% mutation coverage.

2) using abs/ror Mutation adequate test sets

- abs/ror Mutation adequate test sets are not mutation adequate for all 4 programs.
- For Find, Strmat1, Strmat2, textfmt each of the 30 abs/ror Mutation adequate test sets give a mutation score of at least 97.67%, 97.60%, 91.48%& 92.73% respectively.

Above observations indicate that x% & abs/ror Mutation adequate test sets suffer just less than 5% loss on average mutation score.

Offutt et. al.(1993) have proposed N-selective mutation which concentrates on omitting the N most plentiful operators .They have used simple and multiple linear regression models to calculate the complexity of mutation testing. They have done experiments on 10 Fortran 77 programs and compared the mutation scores for 2-selective, 4-selective, 6-selective mutations against non selective mutation. Data indicates that the test sets that were 100% adequate for 2-selective mutation were also almost 100% adequate for non selective mutation. This shows that the 2 most plentiful operators (SVR and ASR) were not orthogonal with all others and hence can be dropped from the list for the cost effectiveness of mutation. For experimenting on 4-selective mutation testing they added SCR and CSR operators to the existing 2 and got favorable results. Even without these 4 mutation operators test sets, had an average non selective mutation score of 99.84%. Extending for 6-selective mutation testing they have omitted CAR and ACR mutant operators and concluded with an average non selective mutation score of 99.71%. All these experiments help us to suggest that all procedural mutation operators suggested by Mothra are not orthogonal.

Offutt et. al (1996) have extended the same experiments for ES-selective, RS- selective, RE-

selective mutation testing and noted the wonderful results. Average Non selective mutation score of ES-selective, RS-selective and RE-selective mutation adequate sets were found to be 99.54%, 97.31% and 99.97% respectively. This again shows that all procedural mutant operators are not always orthogonal.

Namin and Andrew (2006) have made use of 7 C applications from siemens for finding out sufficient mutant operators. They have used proteum with 108 mutation operators for C for generating mutants in the experiment. They have used 3 statistical methods(All Subsets Regression(SUB), Elimination Based Correlation technique(EBC), Cluster Analysis(CA)) of variable reduction for the purpose of reducing the number of mutation operators in the set of sufficient operators. Namin et al.(2008) have extended the work done in previous paper and suggested only 28 mutation operators as sufficient as opposed to 108 in proteum and thus having an overall cost reduction and quality enhancement This again shows that the actual number can be reduced to achieve almost same functionality with less number of mutation operators and hence are not orthogonal.

Orthogonality of Object Oriented Mutation Operators

Lee, Ma and Kwon (2004) have worked on orthogonality of class mutation operators. They have done experimental evaluation of effectiveness of class mutation operators (proposed by Ma et al.(2002)) with two java applications using Mujava tool for automatically generating and executing mutants. The number of mutants generated for both of the applications was very small. Only 5 mutation operators were applicable for the first application and they produced only 54 mutants in total. Only 3 mutation operators have generated mutants for second application and they have produced 82 mutants in all. Mutation scores for both applications show that mutation operator adequate test set killed only those mutants generated with the operator itself, except in some exceptions. All these results show maximum possibility of class mutation operators being orthogonal.

Orthogonality of Declarative Mutation Operators

In declarative paradigm work has been done on SQL mutation operators by Tuya et al.(2007) . A set of SQL mutants based on features present in a conceptual model of the database schema has been presented by Chan and Cheung (2005). Mutation operators for SQL injection vulnerabilities (SQLIVs) have been proposed by Shahriar (2008). Automatic Generation of Mutants of SQL Queries is also possible with the help of a tool named SQL Mutation as proposed by Tuya, Suarez-Cabal, Riva (2006). Tuya et al.(2007) have performed experiments on SQL Queries to explore some method to lower the cost of mutation testing on SQL transactions. They have used SQL mutation tool for automatic generation of mutants and collected SQL test suite from NIST conformance Test suite software web pages. They have performed tests for two levels: Entry SQL and Transitional & Intermediate SQL. They have used 5 different steps to complete the test sets to achieve 100% mutation scores. They have tried to find out some selective mutation operators in order to reduce the cost of testing. They have performed experiments on selective mutations according to mutant category as well as mutant type. They were not able to find out any mutation operator or category by dropping which mutation adequacy criteria doesn't suffer and reduces the cost. That shows that all of the mutant categories as well as individual operators explore some different type of faults in SQL Queries. Hence none of them performs the same job as done by other. Therefore we can conclude that SQL mutation operators are orthogonal in al most all cases.

Conclusion

Mutation operators in different paradigms exhibit different characteristics. Procedural mutation operators are supposed not to be orthogonal. Some of them reveals the same type of errors as by others. Therefore the set of procedural mutation operators can be reduced by examining the orthogonality of the operators. And hence the cost of using mutation on procedural languages can be reduced a lot. In the case of class mutation operators experiments have shown that only few operators are applicable to different

classes in different situations and they further have produced limited mutants. Almost all operators are capable of detecting some different type of faults existing in the programs, so it isn't possible to reduce the list and hence they exhibits the property of orthogonality. Also in declarative paradigms mutant operators for SQL language are found to be orthogonal. Orthogonality is a feature of programming languages that can be used very well to judge the size of mutation operators. This feature can be used successfully to reduce the cost of using mutation testing in different paradigms.

References

1. Chan, W.K. , Cheung, S.C. And Tse, T.H. (2005): Fault-Based Testing of Database Application Programs with Conceptual Data Model In Proc. of the Fifth International Conference on Quality Software, 2005. IEEE Computer Society Press, Los Alamitos, California, 187-196.
2. Gupta, D. (2004): What is a Good First Programming Language? In Cross Roads Homepage Archieve, 10(4), ACM, New York.
3. Lee H.J., Ma Y.S. and Kwon Y.R.(2004): Empirical Evaluation of Orthogonality of Class Mutation Operators. In Proceedings of 11th Asia-Pacific Software Engineering Conference (APSEC'04).
4. Mathur, A.P. and Wong, W.E.(1995): Reducing the Cost of Mutation Testing: An Empirical Study. *The Journal of Systems and Software*, 31(3), 185-196.
5. Namin, A.S. and Andrews, J.(2006): Finding Sufficient Mutation Operators Via Variable Reduction. In Proc. of 2nd Workshop on Mutation Analysis.
6. Namin, A.S, Andrews, J.H. and Murdoch, D.H.(2008): Sufficient Mutation Operators for Measuring Test Effectiveness. In ICSE '08: Proceedings of the 30th International Conference on Software Engineering, New York, NY, USA. ACM, 351-360.
7. Offutt A.J., Rothermel G. and Zapf C.(1993): An Experimental Evaluation of Selective Mutation. Fifteenth International Conference on Software Engineering, Baltimore Maryland, USA, 100-107.
8. Offutt A.J., Rothermel G., Untch R.H. and Zapf C.(1996): An Experimental Determination of Sufficient Mutant Operators. *ACM Transactions on Software Engineering Methodology,* 5(2), 99-118.
9. Shahriar, H.(2008): Mutation-Based Testing Of Buffer Overflows, SQL Injections, And Format String Bugs. Masters of Science Thesis, School of Computing, Queen's University, Kingston, Ontario, Canada.
10. Tuya, Suarez-Cabal, M.J., Riva, C.D.L. (2006): SQL Mutation A Tool to Generate Mutants of SQL Database Queries In Second Workshop on Mutation Analysis, Nov 2006, 1.
11. Tuya, Suarez-Cabal, M.J., Riva, C.D.L. (2007) : Mutating Database Queries. *Information and Software Technology,* 49(4), 398-41.