International Journal of Information Processing, 7(1), 30-38, 2013
ISSN : 0973-8215
IK International Publishing House Pvt. Ltd., New Delhi, India

A Statistical Method for Generating Test Sets with a Given Coverage Probability

C Selem[a], E A Schmitz[a], A J Alencar[a], J V Doria[a], F Protti,[b]

[a]The Tércio Pacitti Institute, Federal University of Rio de Janeiro (UFRJ), Fundão Island, Rio de Janeiro, RJ 21941-916, Brazil, Contact: eber@nce.ufrj.br

[b]Fluminense Federal University (UFF), Computer Science Institute, Niterói, Rio de Janeiro - Brazil

This paper presents a method that uses a small set of execution samples to select a minimal set of execution paths, which have the property that its coverage probability is above a required confidence level. From this path set, a natural language specification of the test case set can be derived. Preliminary experimental results show that it is not only simple to be applied but generates a reliable test case set. The greatest benefit, however, is the use of a minimal set of test cases, which reduces the computational effort to guarantee that the resulting test case set will exercise all program paths which, as a whole, have a probability of occurrence above a required confidence level.

Keywords: Coverage Probability, Statistical Method

1. INTRODUCTION

The essence of the "basis path testing", first proposed by McCabe [1] in 1976 is the representation of the source code as a type of graph. The Control Flow Graph (CFG) is a directed graph where nodes represent computations and edges the transfer of control. From this graph, a set of basis paths are identified and a test case is generated for each one. This technique generates a minimal set that covers all non-redundant execution scenarios for the module. The evaluation of decision and loop nodes being dependent on the unknown state of variables at the time of execution implies that each traversed path element of the set can be associated with an execution probability. Since the coverage obtained from the basis test paths set approach is probabilistic, the use of this technique can potentially generate a test paths set which has little chance of being executed in practice.

When the objective is to define a test paths set such that the coverage probability is above a required confidence level, we must take a different approach. The aim now is to find the set of test cases that guarantee a minimum required probability of being executed. If the probability distribution of the execution paths were known, it would be a simple matter of selecting the smallest set that attains to the required level of confidence. Since this distribution is unknown, we could use instead, the probability distribution of the paths obtained by running the program a very large number of times, let's say N, which would converge to the real distribution of execution paths. But this solution would require an amount of computational effort equal to the testing the software itself.

This paper presents an efficient method to find the test case set through the selection of a minimal set of paths, that satisfies the coverage probability, by using the combination of execution samples of size n ($n \leq N$) with a generative algorithm, in such a way that this set can be obtained in a very efficient way. The method has been tested on a set of programs and the results show that the set of paths produced is statistically similar to the one obtained by running a very large sample.

Table 4
Experimental Results

Program	Activity Nodes	Junction Nodes	Decision Nodes	Loop Nodes	Reject H_0?
inAddress	5	1	2	1	Yes
c8up	4	2	1	1	Yes
comb	6	1	2	1	No
noLetters	4	2	1	1	Yes
upper	5	1	1	1	Yes
factorial	4	1	0	2	Yes
range	6	4	3	1	Yes
snacks	7	1	4	1	Yes
avgAge	7	3	0	3	Yes
noVowsCons	9	1	4	1	Yes
bisection	8	2	2	1	Yes
binaryNum	5	1	2	1	Yes
primeGen	6	2	2	2	Yes
rot13	6	1	4	1	Yes

Table 5
Programs' Execution Time

Program	25 executions (in nanoseconds)	1000 executions (in nanoseconds)
inAddress	9	180
c8up	Next to 0	115
comb	9	375
noLetters	Next to 0	173
upper	9	78
factorial	9	234
range	9	750
snacks	9	84
avgAge	9	691
noVowsCons	9	103
bisection	9	972
binaryNum	9	48
primeGen	9	936
rot13	868	38554

does not use random inputs; instead, the values of the inputs are known, and several possibilities are combined.

5. CONCLUSIONS

This paper presents a computationally efficient method to generate a test set that guarantees a probabilistic coverage above a required confidence level. The method consists of obtaining an approximation of the probability distribution of the set of execution paths using a combination of a sampling procedure, which provides a low cost of execution, and a generative algorithm based on the probabilistic analysis of a special form of control flow graph. The evaluation tests gave us some confidence that not only the method is simple to use, but also provides a good approximation when comparing $\mathcal{P}_A$ (the set of paths which is the outcome of the method) and $\mathcal{P}_L$ (a large sample of the real set $\mathcal{P}$ of execution paths). The greatest benefit of our method, however, is the use of a minimal set of test cases, which reduces the computational effort to guarantee that the program reliability is above a required level of confidence.

REFERENCES

1. Thomas J McCabe. A Complexity Measure, *IEEE Transactions on Software Engineering*, 2(4):308-320, December 1976.
2. V C Barbosa, F M L Ferreira, D V Kling, E Lopes, F Protti and E A Schmitz. Structured Construction and Simulation of Nondeterministic Stochastic Activity Networks, *European Journal of Operational Research*, 198(1):266-274, October 2009.
3. Paul G Hoel. Introduction to Mathematical Statistics, *John Wiley and Sons*, California, USA, 1966.

4. T B Arnold and J W Emerson. Nonparametric Goodness-of-Fit Tests for Discrete Null Distributions, *The R Journal*, 3(2):34-39, December 2011.
5. J A Whittaker and J H Poore. Markov Analysis of Software Specifications, *ACM Transactions on Software Engineering and Methodology*, 2(1):93-106, January 1993.
6. E Goldberg and Y Novikov. BerkMin: a Fast and Robust SATSolver, *in Proceedings of Conference on Design, Automation and Test in Europe*, 142-149, April 2002.
7. J Callahan, F Schneider and S Easterbrook. Automated Software Testing using Model-checking, *in Proceedings of 1996 SPIN Workshop*, August 1996.
8. K Burr and W Young. Combinatorial Test Techniques: Table-based Automation, Test Generation and Code Coverage, *in Proceedings* 7^{th} *International Conference on Software Testing, Analysis, and Review*, October 1998.

C Selem is an M.Sc. Student with the Federal University of Rio de Janeiro (UFRJ), Brazil. Her research interests include Software Development Methodologies and Software Engineering Testing Methodologies.

E A Schmitz is a Professor of Computer Science with the Federal University of Rio de Janeiro (UFRJ). He holds a B.Sc. in Electrical Engineering from the Federal University of Rio Grande do Sul (UFRGS), an M.Sc. in Electrical Engineering from the Federal University of Rio de Janeiro and a Ph.D. in Computer Science from the Imperial College of Science Technology and Medicine, London, England. His research interests include Software Development Modeling Tools, Business Process Modeling and Stochastic Modeling.

F Protti is a Professor of Computer Science with the Fluminense Federal University of Rio de Janeiro (UFF). He holds a B.Sc. in Computing Science from the University of So Paulo and an M.Sc. and a D.Sc. in System Engineering and Computer Science from the Federal University of Rio de Janeiro. His research interests include Algorithms and Graph Theory.

A J Alencar is a researcher with the Federal University of Rio de Janeiro (UFRJ), Brazil. He received his B.Sc. in Mathematics and M.Sc. in System Engineering and Computer Science from UFRJ. He holds a D.Phil. in Computer Science from Oxford University, England. His research interests include Economics of Software Engineering, IT Strategy and Risk Analysis.

J V. Doria Jr. is an M.Sc. Student with the Federal University of Rio de Janeiro (UFRJ), Brazil. His research interests include Software Development Methodologies and Economics of Software Engineering.