Test Case Generation for Boolean Expressions by Using Set and Cell Covering

B.Prasanna[#1], Dr. P. Radhika Raju[*2], Dr. A. Ananda Rao[#3]
Department of Computer Science & Engineering, JNTUA College of Engineering, Ananthapuramu, India
[1] M.Tech Scholar
[2] Ad-hoc Assistant Professor
[3] Professor

***Abstract*— Boolean expressions focus primarily on specifications and find the faults and are introduction to finding the faults based on the test techniques. The paper presents different techniques of fault-based testing. It acknowledges that the techniques vary in their fault detection capacities and test suite development and the topological structures of Boolean expression defects including single and double LIF, LOF and LRF defects in K-maps are officially proven to be defined as ILP for the detection issue. It turn decreases the time to produce the test case instances considerably. Also considered were the various techniques for dealing with limitations in Boolean Expression, Minimal Fault Detection Test Suites, Reducing the size of the logic test set using ILP, The logical mutation strategy used by Cell Covering and Boolean Expressions for Test Generation. The fundamental algorithms and fault categories are these strategies are used for formulating their performance. Finally this paper describes the use of Boolean expressions (K-maps) for specifying the requirements in order to detecting the faults and also used to reduce the time and find the faults.**

***Keywords*— Boolean Expressions, K-map, Fault-Based Testing, Test Suite, BOR.**

I. INTRODUCTION

Software Testing aims for detecting errors in a system or program. Increasing the software dimension and complexity has made testing software a difficult activity. The objective of software testing is determine the errors; Basically for checking the test cases it consumes large amount of time so particular exploration and different ways have to be considered to improve the efficiency and effectiveness of test instances.

Test case designing is the important factors that are influenced by the cost as well as the coverage of testing. The cost of testing is depends up on size of test cases and coverage of testing on fault detection capabilities. Many researchers are directed at performing the high efficiency by determining the suitable test instances and reducing the cost of testing. For logical phrases as well as model predicates, these are used to define certain circumstances of requirements. The test cases are generates for Boolean expressions and these are used to detecting faults in the programs.

Many fault-based testing methods have suggested by different scientists to define the test suites which are based on the Boolean expressions; in addition the test instances are found by using methodologies which can ensure that some kinds of the faults are detected. Basically test case is the process of producing the test suites for a particular system. This concept is used to detect faulty conditions from programming languages i.e., java or C/CPP or any other programming language by generating test cases and by building KMAP. These faulty conditions can be given by programmer by mistake or by intention. A condition (Boolean expression) will be parsing to find out all possible conditions by adding one literal or by omitting one literal and then evaluating that condition to predict faulty results. In this Expressions AND, OR and NOT operators are used.

This article uses Cell Covering models for Boolean Expressions by considering the Test Case Generation approaches for. Boolean expressions are commonly discovered within programs in logical condition and identify which models are complicated conditions. This article examines test instances which are created from Boolean expressions which are targeted by the particular Classes of faults and the test suites are reduced for full testing. In this article, readers are presumed to be closer to Boolean explanations and terminologies.

II. TYPES OF FAULTY DETECTION

There are various types of faulty techniques namely,

1. Cause effect graph
2. Branch Operator Strategy (BOR)
3. BOR+MI
4. MUMCUT
5. Modified Condition/Decision Coverage (MCDC).

MCDC performance it's much better for all types of mistakes than BOR, depending upon the types of defective detection methods. The size of the test cases which is also similar with BOR.MUMCUT detecting all MI-detected faults and the test produced is a sub-set of MI-generates the test sets and the size of the test suite is much lower.

III. RELATEDWORK

A. Gargantini ***(2011)****[*4] Proposed The limitations among the factors contained in it should also be considered when testing a Boolean expression. Constraints the interdependence of the model among the terms. The author provided three methods to address such limitations:(1) ignores them in the form of test generation and subsequently remove invalid tests; (2) include them in the joint speech and subsequently remove invalid tests again, and (3) The sample generation phase are considered so that valid tests can only be generated from the beginning.

G. Fraser and A. Gargantini (2011)[5] Described a technique that produces test instances straight from possible faults of the expression, ensuring the detection of faults of any selected class. This strategy does not requires Boolean expressions To be in standard disjunctive form (DNF) as opposed to many prior criteria, but enables phrases in any format using any intentional classes of fault. Sample paragraph Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper. Test predicates are produced in Figure 1 provided a set of Boolean sentences and fault classes (1) A SAT or SMT (Satisfied Modulo Theories) solver is used by the test suite generator (2) to search for each of these test predicates, a model may be used as a test case if a model exists.

*G. Kaminski and P. Ammann(2011)[*6] Minimum Conjunctive Normal Form (CNF) was given, along with general Boolean expressions. A determination is generated with Minimal-MUMCUT at the level of the literal elements and the conditions of which constituent criteria are viable and therefore essential. Some of the researchers found that Minimal-MUMCUT decreases Set test size detection regardless of Format for predicate. (1) indicates a powerful arrow from the source error to a destination error; an associated destination fault will also be detected (2)again demonstrates that a matching target fault will also be detected if a test will detect a source error.

Lian Yu and Wei-Tek Tsai(2018) [11] Discussed characterizes Boolean flaws as modifications in Topological structures in K-map fields that are declining and/or expanding. A cell cover is a collection of K-map neurons (test cases) to cover failure areas to ensure detection of faults. It can be developed as an ILP problem to minimize cell coverage. The original Problem by evaluating the constraint matrix constructions, which considerably decreases the time required to execute the ILP. To solve these complicated Boolean expressions by matching the cell coverage issue, use an effective rough algorithm with a tight theoretical boundary. The optimal technique and strategy for defining test cases are combined into a hybrid technique Method for defining ILP relaxation fraction assessment test cases.

A. Boolean Expressions Solving Procedure

Boolean Operator testing Strategy: If T(E) meets the BOR testing strategy for E, the test set is said to be T(E) is a BOR test set for E. If E is a straight-forward Boolean expression,{ (t),(f)} will give the minimum BOR test set for E. If E is an expression of Boolean compounds, then E can be represented as E1 op E2, where op is either or+, and E1, E2 is either straight or Boolean. Seven sample cases chosen for N7 implementing approach(t, f, t, t)(f, t, t)(f, f, t, t)(t, f, t, f),t, f, f, t)(t, f, f, f, f),f)(f, f, f) } to figure 1.

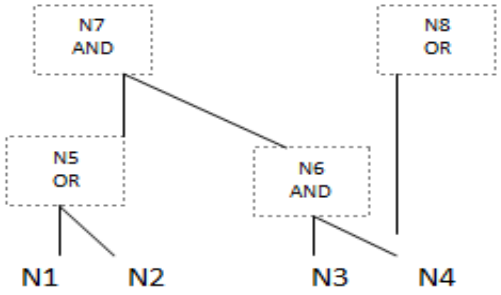

Fig.1: A Cause Effect Graph

Example: two lowest test sets created by applying BOR Strategy.
St(N7)={(t, f, t, t)(f, t, t, t) }
Sf(N7)={(f, f, t, t)(t, t, f) or { f, f, t, t)(f, t, t),f)(f, t, t, f)(f, t, t, f) }

B. BOR+MI: The approach of BOR+MI and MI have a similar capacity to detect faults.
*Algorithm for BOR+MI***:** The approach of BOR and MI has similar capacity to detect faults.
*Step 1***:** separation Boolean expression *BE* into equally singular components.
*Step 2***:** Create test cases using *BOR* for every singular component.
*Step 3***:** Create test cases using *MI* for non singular components.
Step 4: Join the conditions generated above

Example: Let BE = a(bc+bd) of the BOR+MI test instances
StBE={(t, t, t, f)(t, f, t, t)
SfBE={(f, t, t, f)(t, t, f)(t, t, f, t)(t)(t, t, t)(t).

C. Minimal-MUMCUT:
The minimal-MUMCUT chooses the Unique True Point UTP I test points so that each missing variable's truth value is covered.
Example: Let E= from+ cd
Set of test cases generated using MUMCUT
By applying MUTP strategy{(t,t,f,t)(t,t,t,f)(f,t,t,f)(t,f,t,t)
By applying MNFP strategy{(f,t,f,t)(f,t,t,f)(t,f,f,t)(t,f,t,f)
By applying CUTPNFP strategy
{(t,t,f,t)(f,t,f,t)(t,f,f,t)(f,t,t,t)(f,t,f,t) f,,t,t,f)}

Most of the above disclosed strategies ' effectiveness is evaluated in terms of their capacity to identify defective mutations.

IV. PROPOSED FRAMEWORK

The suggested framework is recognized in K-Map. The K-map is a straight-forward technique of simplifying and minimizing the Boolean expressions. It is faster and more efficient than other Boolean expressions simplification techniques. By using this method we can decrease K-Map construction time and reducing the cost too.

It also detects the topological constructions of Boolean expressions flaws Single and double faults included these topological structures of LIFs, LOFs and LRFs in K-maps officially proven to be formulated as ILP for the detection issue. That in turn decreases the time considerably and generates test instances for Boolean expressions as well.

V. IMPLEMENTATION

The project of implemented is to detect faulty conditions from programming languages such as java or C/CPP or any other programming language by generating test cases. In this project as part of extension is used normal and parallel processing technique from java multi thread executors which will use multi core processing speed to execute given task. By using this technique we can reduce K-Map building time.Now click on 'Build K-Map with Parallel Multi Processor' button to build K-Map with parallel technique and to get below screen.

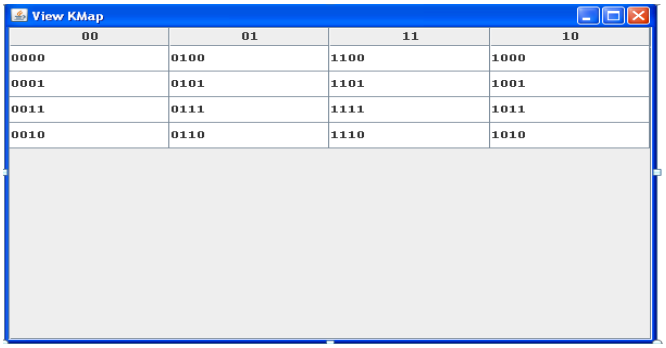

Fig.2: Build K-Map with Parallel Multi Processor

Now click on 'Normal & Parallel Execution Time Graph' button to get execution time for both normal and parallel techniques

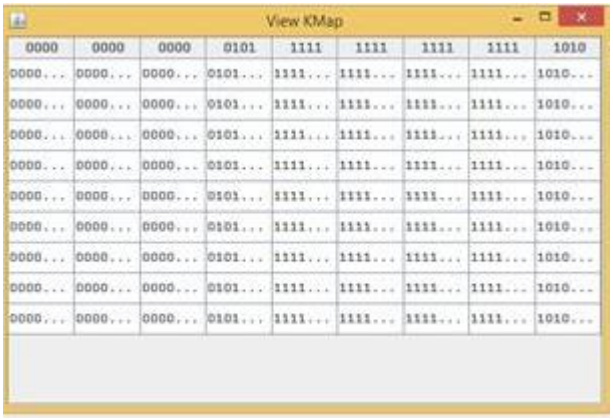

Fig.3: Normal & Parallel Execution Values

VI. EXPERIMENTAL RESULTS

The main observation of experimental results is the computation time with respect to performance. In below graph x-axis represents total test cases and cell covering and y-axis represents count. As presented in the fig 1, different number of techniques is presented in the horizontal axis while the vertical axis showed computational time in nanoseconds. It is evident that the computation time and test sizes with existing system models like test cases size and cell covering.

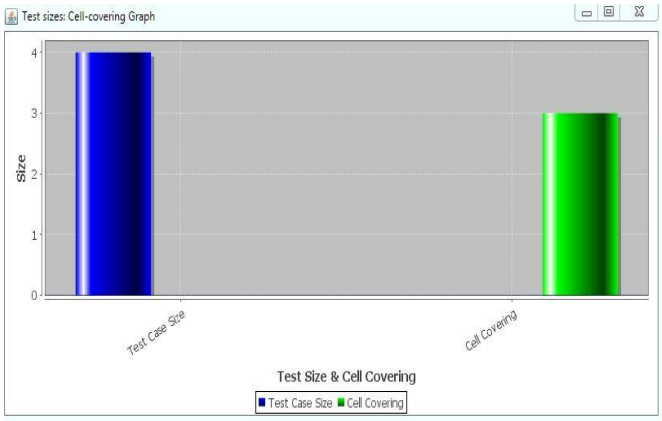

Fig.4: System Time Comparison of Test Sizes and Cell Covering .

In below graph x-axis represents technique name and y-axis represents execution time for building k-map using both normal and parallel technique

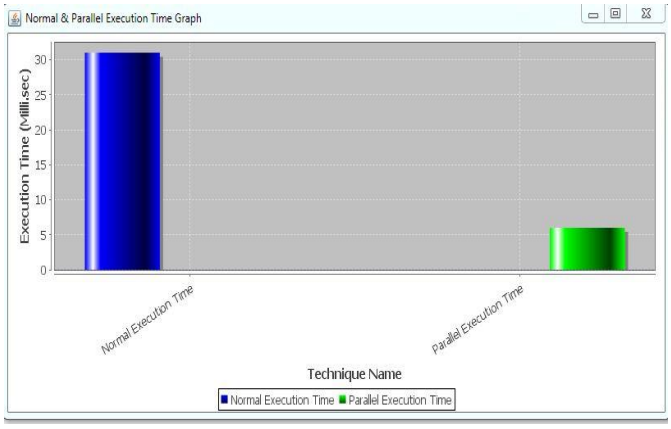

Fig.5: The Proposed System with Using both Normal and Parallel Technique

VII. COMPARISON ANALYSIS

This article, readers are presumed to be acquainted with Boolean expressions notations and terminologies. The purpose of this study is to present such methods in one location and provide a foundation for comparison between these methods. To help Boolean Expressions, fault-based techniques, many distinct methods were planned, in a wide range of literature across a broad range of areas and regular allocations, this has been stated. The comparison of this survey research was assessed, especially in the literature on software testing and maintenance of software. It represents a comparison of Boolean expression techniques of speed, cost and quality measures. According to

TABLE I
COMPARISON OF SPEED, COST AND QUALITY BOOLEAN EXPRESSIONS FAULT TECHNIQUES

TECHNIQUES	SPEED	COST	QUALITY
Disjunctive Normal Form (DNF)	Only 3% of these predicates were unique in five or more	high	Detection of a fault of 1.30 % of the size required. If the feasibility is not considered.
SMT	the test case Speed up by 40 percent test and the process of generating data.	high computational	Better quality.
Boolean expression testing (BET)	high	low	high

This comparison the Boolean expression by cell covering techniques performs better prediction than other existing techniques.

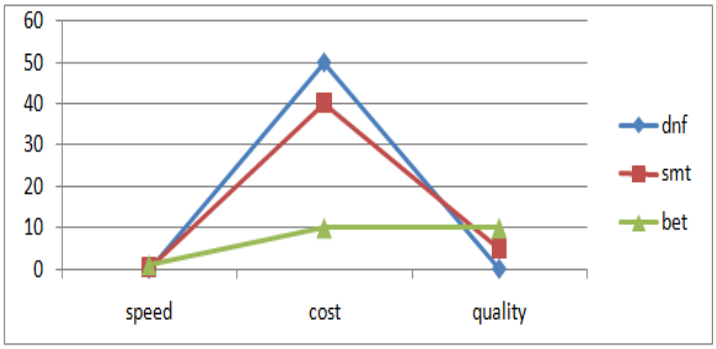

Fig.6: The Graph Measurements of Speed and Cost and Quality.

VIII. CONCLUSION

This article provides a short study of Cell Covering's discussion of Test Case Generation with different categories for Boolean Expressions. Many of the Published fault class study was based on empirical evidence, empirical assessment of Boolean expressions and an approach based on fault detection capacities was conducted. Boolean expressions from literature based on fault-based assessment, different efficiency and efficiency of testing methods. Logic Mutation expressions were produced by creating syntactic change based on specific type of fault from the specified Boolean expressions. The findings were in favor of cell-covering all-fault class detection method, but the size of the test set is big. Originally intended to detect missing/extra negation operators, Boolean expression approximate method; therefore, it does not ensure the detection of other faults. The further research improved and extended the Depth First Searching algorithm for the Boolean expressions method to detect the assessment of fault circumstances can make the algorithm more effective.

IX. REFERENCES

[1] K.C. Tai. Theory of Fault Based Predicate Testing for Computer Programs, IEEE Transactions of Software Engineering, vol 22, no 8, pp 552-562, 1996

[2] K.C Tai. M.A Vouk., A. Paradkar., Lu P. , "Predicate Based Testing," IBM Systems Journal, Vol 33 (3), p 445, 1994

[3] M. A. Vouk, K. C. Tai, and A. Paradkar. Empirical Studies of Predicate-based Software Testing. In 5th International Symposium on Software Reliability Engineering, pages 55–64. IEEE, 1994.

[4] A. Gargantini, “Dealing with constraints in Boolean expression testing,” in Proc. 3rd Workshop Constraints Soft. Testing Verification Anal., Mar. 25, 2011, pp. 322- -327.

[5] G. Fraser and A. Gargantini, “Generating minimal fault detecting test suites for Boolean expressions,” in Proc. 3rd Int. Conf. Soft. Testing Verification Validation Workshops, Apr. 2010, pp. 37–45.

[6] G. Kaminski and P. Ammann, “Reducing logic test set size while preserving fault detection,” Soft. Testing, Verification Rel., vol. 21, pp. 155–193, 2011.

[7] 21] G. Kaminski, U. Praphamontripong, P. Ammann, and J. Offutt, “A logic mutation approach to selective mutation for programs and queries,” Inf. Soft. Technol., vol. 53, pp. 1137– 1152, 2011.

[8] Godefroid, P., Levin, M. Y., and Molnar, D. (2012) SAGE: White box fizzing for security testing. Commun. ACM, 55, 40-44.

[9] Peleska, J. (2013) Industrial-strength Model-Based Testing - state of the art and current challenges. In Petrenko, A. K. and Schlinglo, H. (eds.), Proceedings Eighth Workshop on Model-Based Testing, MBT 2013, Rome, Italy, 17th March 2013, EPTCS, 111, pp. 3-28.

[10] P. Arcaini, A. Gargantini, and E. Riccobene, “How to optimize the use of SAT and SMT solvers for test generation of Boolean expressions,” Comput. J., vol. 58, pp. 2900–2920, Jan. 21, 2015.

[11] Lian Yu and Wei-Tek Tsai, "Test Case Generation for Boolean Expressions by Cell Covering", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 44, NO. 1, JANUARY 2018.

[12] Kaminski G, Ammann P. Using logic criterion feasibility to reduce test set size while guaranteeing fault detection. Proceedings of the 2nd International Conference on Software Testing, Verification and Validation, Denver, CO, April 2009; 167–176.

[13] Kaminski G, Ammann P. Using logic criterion feasibility to reduce test set size while guaranteeing double fault detection. Proceedings of the Mutation Workshop at the 2nd International Conference on Software Testing, Verification and Validation, Denver, CO, April 2009.

[14] G. Kaminski and P. Ammann, “Applications of optimization to logic testing,” in Proc. Softw. Testing, Verification Validation Workshops, 2010, pp. 331–336.