International Journal of Computer Applications (0975 – 8887)
Volume 172 – No.7, August 2017

Approach for Minimization of Test Cases from Decision Table Generated from Cause Effect Graph

Monika Agrawal
Assistant Professor, Arya Group of Collages
Kukas- Rajasthan

Usha Badhera, PhD
Assistant Professor, Banasthali Vidyapith.
Jaipur, Rajasthan

ABSTRACT
Cause effect Graph focuses on modeling dependency relationships among input conditions of program known as causes, and output conditions known as, effects. The relationship among input conditions and output conditions is expressed pictorially in terms of cause-effect graph. Cause effect Graph is basically hardware-testing technique adapted to software testing by researchers. This paper proposes the details about the techniques to generate test cases from given cause-effect graph. The objective of this paper is to get the minimum number of test cases to find maximum errors.

Keywords
Input, output, effects, test cases, errors, cause, cause effect graph.

1. INTRODUCTION
Testing involves set of operation carried out to assess some prospect of a piece of software; Software testing is a process of executing programs to detect faults. The quality of software can be assured by effective testing. The primary goal of software testing is to identify that software confirm to the requirements. Testing is designed to check if the system behaves as desired, which gives confidence about the correctness of software. In present scenario, society is gradually becoming dependent on software. The complexity of the software has grown, and the constraints of budget and time on software development process have increased. These three factors complexity, cost and time made it practically impossible to guarantee that software is perfect. To reduce cost and time of testing , it is required to generate less number of test cases which has more chances of finding errors. In this study one of the Black box Testing technique, Cause Effect graph is considered to generate fewer test cases with high probability of finding errors.

Preliminary and Previous work Cause-Effect Graphing is basically a hardware testing technique adapted to software testing by Elmendorf [1973] and further developed by Nursimulu et al. [1995] Myers [1979], Myers et al. [2011], Tai [1993] and Paradkar et al.[1997]. It focuses on modeling dependency relationships among program input conditions known as causes, and output conditions known as, effects. The relationship is expressed visually in terms of cause-effect graph. The graph is a visual representation of logical relationship among inputs and outputs that can be expressed as a Boolean expression. One approach to test generation was to consider all possible combinations of causes of the CEG, which is exhaustive in nature but impractical as the test cases generated are exponential function of number of causes in the CEG. A practical test generation algorithm for CEGs was described by Myers [1979] which is referred to as algorithm CEG_Myers. Strengths and weaknesses of Myers approach have been investigated by Nursimulu *et al.* [1995]. Myers process of creating decision table is inconsistent and ambiguous, other researchers Mathur [2008] and Srivastava *et al*. [2009] have given algorithm for creating decision table from cause effect graph for generation of test cases.

Table 1. Notations for Cause Effect Graph

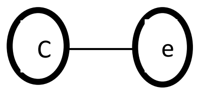
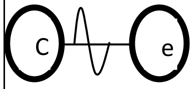
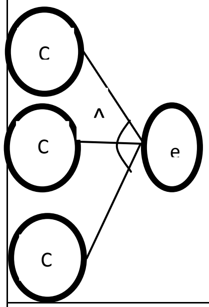
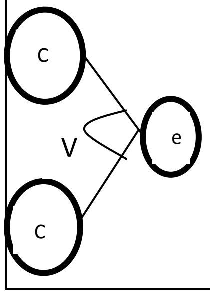
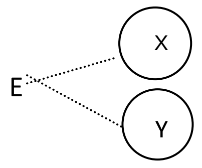
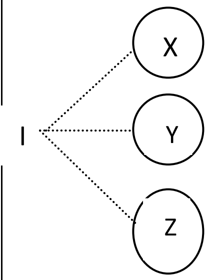
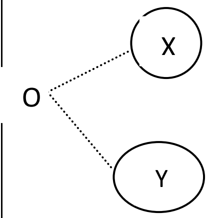
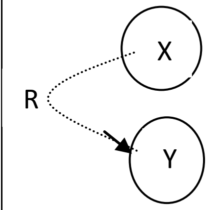
Notation	Definition	For making e true
	C implies e	**C** | **e** 1 | 1
	C not implies e	**C** | **e** 0 | 1
	e when c1 and c2 and c3	**C1** | **C2** | **C3** | **e** 1 | 1 | 1 | 1
	e when c1 or c2	**C1** | **C2** | **e** 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1

Table 2. Constraint symbol

Constraint Symbol	Definition	Table
	The "E"(Exclusive) constraint states that only one of the causes X and Y can be true**.**	**X** | **Y** 0 | 0 0 | 1 1 | 0

	The “I” (Inclusive (at least one)) constraint states that at least one of the causes X, Y and Z must always be true (X, Y and Z cannot be false simultaneously).	X Y Z 1 1 1 1 0 0 0 1 0 0 0 1
	The “O”(One and Only One) constraint states that one and only one of the causes X and Y can be true.	X Y 0 1 1 0
	The “R”(Requires) constraint states that for cause X to be true, than cause Y must be true. In other words, it is impossible for cause X to be true and cause Y to be false**.**	X Y 1 1 0 0

2. PROPOSED APPROACH FOR GENERATING TEST CASES

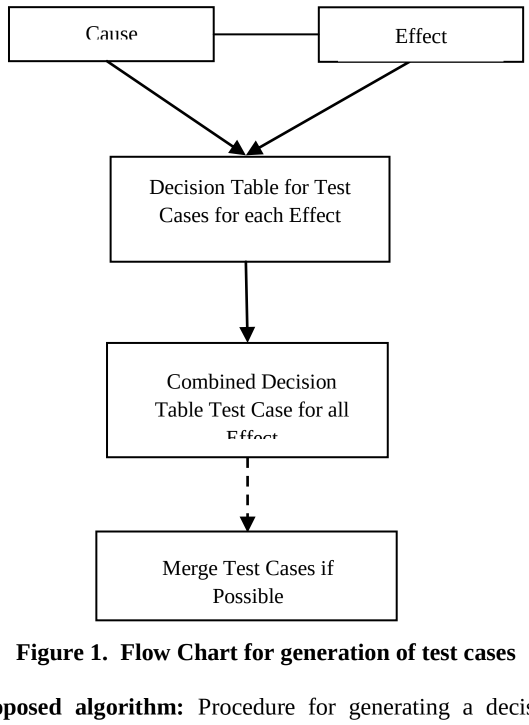

Cause
Effect
Decision Table for Test Cases for each Effect
Combined Decision Table Test Case for all Effect
Merge Test Cases if Possible

Figure 1. Flow Chart for generation of test cases

Proposed algorithm: Procedure for generating a decision table from cause-effect a graph.

Input: (a) A template containing causes C_1, $C_2 \ldots C_a$, effects Ef_1, $Ef_2 \ldots Ef_b$, their relationship & constraint.

Output*:* A decision table DT containing N=a + b rows and M columns, where M depends on the relationship between the causes and effects as captured in the cause-effect graph.

Procedure:[Figure 1] DT_FROM_CEG & initialize DT to an empty decision table.

Step1: Analyze the causes & effects from the given cause-effect graph.

Step2: Execute following steps for i = 1 to b.

2.1- Select the next effect to be processed, Let e = $Ef_{i.}$

2.2- Get the combination of Causes (C_1,C_2,...etc.) such that effect ***e*** to be true[Table 1].

2.3- Apply all the constraints and remove infeasible test cases[Table 2].

2.4- Add these changes to the individual decision table.

Step3: Merge all individual decision table to final decision table:-

if (an effect is independent of some set of causes i.e some set of causes have no impact on a particular effect) - then reduce the number of columns in the decision table by merging the columns(representing set of causes) for that particular effect with the set of columns for some other effect.

else- Add individually

End of Procedures: DT_FROM_CEG

3. EXAMPLE

Proposed algorithm is applied on cause effect graph as in Figure 2

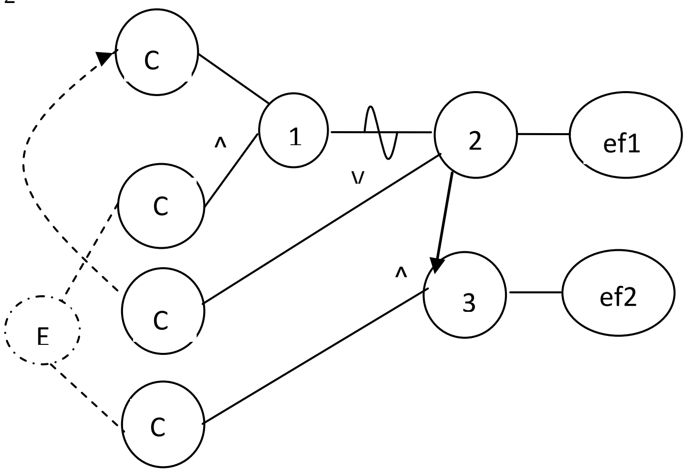


Figure 2. Problem Cause-Effect Graph

Input: (a) A template containing causes C_1,C_2,C_3,C_4, effects ef1, ef2 and constraint are exclusive and requires (a=4,b=2)[see Figure 2]

Output*:* A decision table DT containing N rows and M columns, where N= a + b (a=4 & b=2) therefore N= 6 . M depends on the relationship between the causes and effects as captured in the cause-effect graph.

Procedure: DT_FROM_CEG & initialize DT to an empty decision table.

Step1: There are 4 causes and 2 effects, **requires** constraint exist between c1 and c3, **exclusive** constraint exists between c2 and c4. Ef1 depends on c1, c2 and c3 and ef2 depends on c1, c2, c3 and c4[In Figure 2].

Step2: Execute following steps for i = 1 to 2 (effect 1 to 2).

2.1- Select the next effect to be processed, Let e = ef1.

2.2- Get the combination of causes (c1, c2, c3, node1, and node2) such that effect ***ef1*** to be true[From Figure 2]

In given example[Figure 2] to **make** effect1 true[using Table 1]: node2 should be true which is dependent on c3 and node 1.

C3	**node 1**
0	**0**
1	**1**
1	**0**

Table 3. For effect1(ef1)

[Using Table 1 & Figure 2]

C1	C2	C3
1	0	0
0	1	0
0	0	0
1	1	1
1	0	1
0	1	1
0	0	1

Step-2.3

After Applying Constraints

[using Table 2]

Table 4. After applying constraints in Table 3

C1	C2	C3
0	1	0
0	0	0
1	1	1
1	0	1

In above all test-cases effect 1(ef1) can be true[Table 3]. But there are some infeasible test-cases ,For removal of infeasible test-cases some constraints applied [Table 4].

2.4- Add these changes [Table 4] to the individual decision table[Table 5].

Table 5. Individual Decision Table for effect 1(ef1)

C1	**0**	**0**	**1**	**1**
C2	**1**	**0**	**1**	**0**
C3	**0**	**0**	**1**	**1**
ef1	**1**	**1**	**1**	**1**

Now e = ef2 (To make ef1 true)

node 3 has to be true which is dependent on node 2 and c4[Figure 2]

Table 6. Table for effect 2 (ef2)[using Table 1 & Figure 2]

C1	C2	C3	C4
1	0	0	1
0	1	0	1
0	0	0	1
1	1	1	1
1	0	1	1
0	1	1	1
0	0	1	1

After Applying Constraints

[using Table 2]

Table 7. For effect 2(ef2) after applying constraints in table 6[Table 2]

C1	C2	C3	C4
0	0	0	1
1	0	1	1

In above all test-cases effect 2(ef2) can be true[Table 6]. But there are some infeasible test-cases ,For removal of infeasible test-cases some constraints applied [Table 7].

2.4- Add these changes[Table 7] to the individual decision table[Table 8].

Table 8. Individual Decision Table for effect 2(ef2)

C1	**0**	**1**
C2	**0**	**0**
C3	**0**	**1**
C4	**1**	**1**
ef2	**1**	**1**

Step3: Merge all individual decision tables[Table 5 & Table 8] to Combined decision table [Table9]:-

Table 9. Combined Decision Table For Effect 1(ef1) And Effect 2(ef2)

C1	0	0	1	1	0	1
C2	**1**	**0**	**1**	**0**	**0**	**0**
C3	**0**	**0**	**1**	**1**	**0**	**1**
C4	**0**	**0**	**0**	**0**	**1**	**1**
ef1	**1**	**1**	**1**	**1**	**1**	**1**
ef2	**0**	**0**	**0**	**0**	**1**	**1**

__C4__ is not affecting the __ef1__ ,so we can merge the causes of __ef2__ with __ef1__ , (column 5 and 6 merge with column 2 and 4 respectively[Table 9])

Table 10. Final Decision Table (M=4 & N= 6)

C1	0	0	1	1
C2	**1**	**0**	**1**	**0**
C3	**0**	**0**	**1**	**1**
C4	**0**	**1**	**0**	**1**
ef1	**1**	**1**	**1**	**1**
ef2	**0**	**1**	**0**	**1**

There are 6 test-cases [Table 9]. In Table 10 we have minimize test -cases(2 test cases).Here number of columns are 4(i.e. M=4) & Number of rows are 6 (i.e. N= a + b = 4+2=6), i.e. test cases are 4 generated finally[Table 10]

4. CONCLUSION

The proposed approach is based on algorithm given by Srivastava *et al*[2009] for creating decision table from cause effect graph which is further considered for generation of test cases.

This method further minimizes number of test cases when there is some set of causes which is not affecting all effects, in that case test cases are merged to reduce number of test cases[Table 10].

5. REFERENCES

[1] Srivastava, Praveen Ranjan, Parshad Patel, and Siddharth Chatrola. "Cause effect graph to decision table generation." *ACM SIGSOFT Software Engineering Notes* 34.2 (2009): 1-4.

[2] Myers, Glenford J., Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[3] Badhera, Usha, G. N. Purohit, and S. Taruna. "Fault Based Techniques for Testing Boolean Expressions: A Survey." *arXiv preprint arXiv:1202.4836* (2012).

[4] Mathur, Aditya P. *Foundations of Software Testing, 2/e*. Pearson Education India, 2008.

[5] Paradkar, Amit, Kuo-Chung Tai, and Mladen A. Vouk. "Specification-based testing using cause-effect graphs." *Annals of Software Engineering* 4.1 (1997): 133-157.

[6] Nursimulu, Khenaidoo, and Robert L. Probert. "Cause-effect graphing analysis and validation of requirements." *Proceedings of the 1995 Conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 1995.

[7] Elmendorf, William R. *Cause-effect graphs in functional testing*. IBM Poughkeepsie Laboratory, 1973.

[8] Burnstein, Ilene. *Practical software testing: a process-oriented approach*. Springer Science & Business Media, 2006.

[9] Ferriday, Cai. "A Review Paper on Decision Table-Based Testing." *Swansea University, CS339-2007* 20 (2007): 952-965.