Generating Test Cases Satisfying MC/DC from BDDs

Faustin Ahishakiye, Volker Stolz, and Lars Michael Kristensen

Western Norway University of Applied Sciences, Bergen, Norway
{faustin.ahishakiye,volker.stolz,lars.michael.kristensen}@hvl.no

1 Introduction

Binary decision diagrams (BDDs) are a canonical form for representing Boolean functions. BDDs have found wide application in many computer aided design (CAD) tasks [7], symbolic model checking [5, 7], verification of combinational logic [6], verification of finite-state concurrent systems [2], symbolic simulation [6], and logic synthesis and optimization. Different operations used for providing BDDs with efficient graph representation include *reduce, apply, restrict, compose* and *satisfy* [2]. In this paper, we investigate a new application area of BDDs for structural coverage analysis. We propose an approach of encoding modified condition decision coverage (MC/DC) using BDDs to obtain a minimum set of test cases. According to the definition of MC/DC [3, 9], each condition in a decision has to show an independent effect on that decision's outcome by: (1) varying just that condition while holding fixed all other possible conditions (Unique Cause (UC) MC/DC), or (2) varying just that condition while holding fixed all other possible conditions that could affect the outcome (Masking MC/DC). MC/DC is required by certification standards such as, the DO-178C [8] in the domain of avionic software systems, as a coverage criterion for safety critical software at the conditions level. It is highly recommended due to its advantages of being sensitive to program structure, requiring few test cases ($n+1$ for n conditions), and its uniqueness due to the independence effect of each condition. If MC/DC is encoded with BDDs, a minimum set of test cases can be deduced from a reduced ordered BDD (ROBDD) based on the shortest paths. In addition, optimization and coupling conditions (for example B and $\neg B$), which are problematic for MC/DC are handled efficiently using BDDs. We present an algorithm which, given a Boolean expression in the form of an ROBDD, constructs a minimum set of test-cases satisfying MC/DC. Results show that our approach is more efficient compared to selecting MC/DC test cases from a truth table. For the rest of this paper the term BDD refers to ROBDD.

2 Minimum set of MC/DC test cases from BDDs

Theorems 1 and 2 are used to check UC MC/DC and Masking MC/DC from the BDD [1]. For both of them, there should be a pair of paths from that condition to the terminal node 1 and 0, but UC MC/DC requires strictly same path through BDD and cannot be achieved with coupling conditions. Let $f(A,B,C) = (A \wedge B) \vee C$ be a Boolean function, the truth table representing all possible MC/DC pairs is given in Table 1a. To achieve MC/DC we only need the set of $n+1$ test cases as shown in Table 1b (one MC/DC pair for each condition).

Theorem 1. *Given a decision D, a pair of test cases of D satisfies Unique Cause MC/DC for a condition C if and only if: 1) both reach C using the same path through BDD(D); 2) their paths from C exit on two different outcomes and do not cross each other (C excluded).*

Theorem 2. *Given a decision D, a pair of test cases of D satisfies Masking MC/DC for a condition C if and only if: 1) both reach C; 2) their paths from C exit on two different outcomes and do not cross each other (C excluded).*

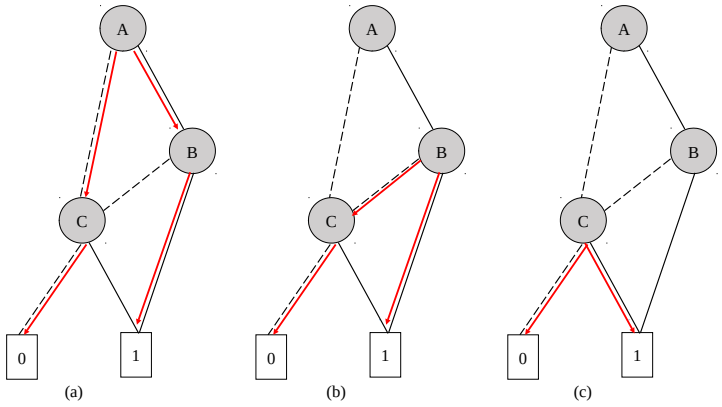

Figure 1: BDD with MC/DC minimum test cases for $(A \wedge B) \vee C$

Nr	A	B	C	f	MC/DC pairs
1	0	0	0	0	
2	0	0	1	1	C(1,2)
3	0	1	0	0	
4	0	1	1	1	C(3,4)
5	1	0	0	0	
6	1	0	1	1	C(5,6)
7	1	1	0	1	A(3,7), B(5,7)

(a) All possible MC/DC pairs

Nr	A	B	C	f	MC/DC pairs
1	0	?	0	0	
2	1	1	?	1	A(1,2)
3	1	0	0	0	B(2,3)
4	0	?	1	1	C(3,4)

(b) Required $n+1$ MC/DC pairs

Table 1: MC/DC pairs for $f(A, B, C) = (A \wedge B) \vee C$

Some conditions may have more than one pair. For instance condition C has three MC/DC pairs, and for example if we pick C(1,2), MC/DC is achieved with five test cases ($\geq n+1$). From the truth table it is not easy to determine which one to choose. Our idea is to take those with the shortest path in the BDD. This is useful especially for a tester, since shorter path means that the tester has to worry about less conditions. The BDD representing $f(A, B, C)$ is shown in Figure 1 (where dashed line:0/false and solid line:1/true). A test case is then simply an assignment of track-values to conditions. For an ROBDD with order $C = [A, B, C]$, $(0, ?, 1)$ denotes the test case with $A = 0$, $C = 1$, and any arbitrary value for B. If we consider the shortest paths from each node to terminal nodes 0 and 1 (shown with red arrows), the minimum set (ψ_{min}) of MC/DC test cases corresponds to $n+1$, equivalently to the Table 1b. Note that the shorter path in the BDD corresponds to the short circuit evaluation "?" in a truth-table where the condition is not evaluated at all. We provide an algorithm that takes a BDD as input and constructs the minimum set of MC/DC test cases. It is summarized in the following steps:

1. Given an ROBDD over the list of variables C. Initialize a set of MC/DC pairs (ψ_i) and the minimum set of MC/DC pairs (ψ_{min}) as empty. The distance from each node u_i to itself is $d(u_i, u_i) = 0$, and to a terminal node is initialized as infinity ($d(u_i, v) = \infty$). The weight of each path σ as $w[\sigma] = \sum_{i=1}^{k} w(u_{i-1}, u_i)$ where the last node $u_i = v$ is the node 0 or 1. To find the shortest path, we compare the distance as $d(u_i, v) > d(u_i, u_i) + w[\sigma]$.

2. While the list C is not empty, find the set of two shortest paths (ψ_i) from each node to 1

and 0 terminal respectively based on Theorem 1 or 2. Only new paths are added to ψ_{min}, to avoid overwriting test cases. For the example in Figure 1, $\psi_1 = \{(0,?,0),(1,1,?)\}$ for node A is added to ψ_{min}, $\psi_2 = \{(1,0,0),(1,1,?)\}$ for node B, only $(1,0,0)$ will be added to ψ_{min}, and $\psi_3 = \{(0,?,0),(0,?,1)\}$ for node C and only $(0,?,1)$ is added to ψ_{min}.

3. If the list is empty, the algorithm returns $\psi_{min} = \{(0,?,0),(1,1,?),(1,0,0),(0,?,1)\}$ which is equivalent to $n+1$ required MC/DC test cases as shown in Table 1. Otherwise, MC/DC can not be achieved. The uncovered conditions will be the remaining variables on C.

In case of strongly coupled conditions (for example B and $\neg B$), our algorithm can still find the minimum set satisfying Masking MC/DC. In addition, BDDs implement directly the optimization for conditions in the decision. For example, $f(A,B,C)=(A \vee \neg A \vee B \vee C) \wedge A$, its BDD representation will contain only one condition (A) because conditions B and C will never be evaluated at all and therefore, they do not appear in the BDD. The optimized representation of BDD helps testers to observe conditions with no effect to the outcome so that there is no need for checking MC/DC for those conditions.

3 Conclusion

This paper presents a new approach of generating test cases satisfying MC/DC from BDDs and provides a new algorithm to deduce the minimum set of test cases satisfying MC/DC based on the shortest path in the BDD. Our results based on small examples show that $n+1$ test cases required to achieve MC/DC for n conditions can be found from the BDD with less overhead compared to selecting them from the truth table. The algorithm can be implemented in Python based on Pyeda BDD library [4]. Moreover, we investigated the formulation of unique cause and masking MC/DC in terms of BDDs. More example results (considering coupling condition and optimization) with the implementation and formal proof of correctness for the algorithm will be provided in an extended paper.

References

[1] Adacore. Technical Report on OBC/MCDC properties. Technical report, Couverture project, 2010.

[2] Randal E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.*, 35(8):677–691, 1986.

[3] John Joseph Chilenski and Steven P. Miller. Applicability of modified condition/decision coverage to software testing. *Softw. Eng. J.*, 9(5):193–200, 1994.

[4] Drake Christopher R. Pyeda: Data structures and algorithms for electronic design automation. In *14th Python in Science Conference (SCIPY)*, 2015.

[5] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[6] Alan J. Hu. Formal hardware verification with BDDs: an introduction. In *Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM*, volume 2, pages 677–682, 1997.

[7] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design*, 1st ed. Springer, 1998.

[8] Frederic Pothon. DO-178C/ED-12C versus DO-178B/ED-12B: Changes and Improvements. Technical report, AdaCore, 2012. Available at https://www.adacore.com/books/do-178c-vs-do-178b.

[9] Leanna Rierson. *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*. CRC Press, 2013.