

A New Test Data Selection Strategy for Testing Boolean Specifications

Noritaka Kobayashi, Tatsuhiro Tsuchiya and Tohru Kikuno
Department of Informatics and Mathematical Science
Graduate School of Engineering Science, Osaka University
E-mail: {n-kobays,t-tutiya,kikuno}@ics.es.osaka-u.ac.jp

1 Introduction

Software testing plays an important role in the process of software development. Usually, exhaustive testing is infeasible because the number of possible test cases is typically prohibitively large. Much research has been aimed at simultaneously achieving high efficacy and reduced cost of testing by selecting appropriate test cases.

In this note, we discuss testing of an implementation intended to satisfy a given specification that is a Boolean formula. Many portions of the behavior of software systems can be represented as a Boolean formula. Conditional state transitions are typical examples.

So far several methods have been proposed for automatically generating test cases [1, 3, 5]. Weyuker et al [5] propose a family of strategies for automatic generation of test cases for Boolean specifications. The empirical results on the fault detecting ability of these strategies are very encouraging. Among the family of strategies, we consider two strategies the most important. The basic meaningful impact strategy (*BMIS*) is the most fundamental strategy. The *MAX-A* strategy is the most powerful strategy, since the strategy guarantees the detection of many types of faults.

In this note, we propose a new test data selection strategy, which also guarantees the detection of all these types of faults. Using an example, we show that the proposed strategy can achieve the same effectiveness as *MAX-A* with much lower cost.

2 Terminology

In this note, 1 and 0 are used to mean “TRUE” and “FALSE”, respectively. We use “.”, “+”, and “-” to represent the Boolean operators AND, OR, and NOT, respectively. Usually, the “.” will be omitted. The set of all truth values ($\{0,1\}$) and the n -dimensional Boolean space are denoted by \mathcal{B} and \mathcal{B}^n , respectively.

We assume that a Boolean specification S is expressed in irredundant disjunctive normal form, that is, $S(\vec{x}) = p_1(\vec{x}) + p_2(\vec{x}) + \dots + p_m(\vec{x})$ where m is the total number of terms in S and p_i ($i = 1, \dots, m$) is the i th term of S . The set of variables in S and the set of variables of p_i are denoted by V and V_i , respectively. A *test case* or *point* for S is an element in the n -dimensional Boolean space \mathcal{B}^n . A point $\vec{x} \in \mathcal{B}^n$ is defined as a *true point* (*false point*) of S , if $S(\vec{x}) = 1(0)$. The sets of all true points and all false points of S are denoted by $TP(S)$ and $FP(S)$, respectively.

A point $\vec{x} \in \mathcal{B}^n$ is defined as a true point of p_i in S , if $p_i(\vec{x})$ evaluates to 1. A point $\vec{x} \in \mathcal{B}^n$ is defined as a *unique true point* of p_i in S , if $p_i(\vec{x})$ evaluates to 1 and

$p_l(\vec{x})$ evaluates to 0 for any $l \neq i$. The sets of all true points and all unique true points of p_i in S are denoted by $TP_i(S)$ and $UTP_i(S)$ ($i = 1, \dots, m$), respectively.

Let $p_i = x_{i1}x_{i2} \dots x_{in_i}$ be the i th term of S where x_{ij} denotes the j th literal in p_i ($j = 1, \dots, n_i$) and n_i is the number of literals in p_i . We use $p_{i,j} = x_{i1}x_{i2} \dots \bar{x}_{ij} \dots x_{in_i}$ to represent the term obtained from p_i by negating its j th literal x_{ij} . A point $\vec{x} \in \mathcal{B}^n$ is defined as a *near false point* of p_i in S , if $p_{i,j}(\vec{x})$ evaluates to 1 and $\vec{x} \in FP(S)$. The sets of all near false points of $p_{i,j}$ in S is denoted by $NFP_{i,j}(S)$.

In the remainder of this section, we consider the Boolean formula $S_1 = \bar{a}bd + \bar{c}d + e$ to illustrate our terminology. This example contains five variables, a, b, c, d , and e . Four points, 10010, 10011, 10110, and 10111, are the true points for $p_1 = \bar{a}bd$. Using decimal notation, these points are represented by 18, 19, 22, and 23. Therefore, $TP_1(S_1)$ is $\{18, 19, 22, 23\}$. Similarly, $TP_2(S_1) = \{2, 3, 10, 11, 18, 19, 26, 27\}$ and $TP_3(S_1) = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31\}$. Unique true points are obtained for each term from these true points as follows. $UTP_1(S_1) = \{22\}$, $UTP_2(S_1) = \{2, 10, 26\}$, and $UTP_3(S_1) = \{1, 5, 7, 9, 13, 15, 17, 21, 25, 29, 31\}$.

Since $p_{1,1} = \bar{a}bd$, $p_{1,2} = abd$, $p_{1,3} = \bar{a}\bar{b}d$, $p_{2,1} = cd$, $p_{2,2} = \bar{c}d$, and $p_{3,1} = \bar{e}$, from the definition of $NFP_{i,j}(S)$, $NFP_{1,1}(S_1) = \{6\}$, $NFP_{1,2}(S_1) = \{30\}$, $NFP_{1,3}(S_1) = \{16, 20\}$, $NFP_{2,1}(S_1) = \{6, 14, 30\}$, $NFP_{2,2}(S_1) = \{0, 8, 16, 24\}$, and $NFP_{3,1}(S_1) = \{0, 4, 8, 12, 14, 16, 20, 24, 28, 30\}$.

3 Testing Model

We consider black-box testing of an implementation of a specification that is a Boolean formula and model the testing process as follows. Let boolean formulas S and F denote a specification and its faulty implementation, respectively. Variables in these formulae represent input parameters that take boolean values. A collection of test cases is referred to as a test suite. A fault will be detected if and only if with a test case, F evaluates to a different value than S . For example, let $S_1 = \bar{a}bd + \bar{c}d + e$ and $F_1 = \bar{a}bd + cd + e$. In this case, \bar{c} is replaced by c . Consider test case 01010. F_1 evaluates to a different value than S_1 for the test case. Hence the fault is detected by the test case. On the other hand, when F evaluates to the same value with S for a test case (for example, 10100), the fault is not detected.

4 Proposed Strategy

BMIS selects test cases such that one point is selected from every non-empty $UTP_i(S)$ and $NFP_{i,j}(S)$. On the

other hand, MAX-A selects all points of every non-empty $UTP_i(S)$ and $NFP_{i,j}(S)$.

A variety of faults models have been proposed so far [1, 4, 5]. These models include TNF, LNF, ORF, TOF, LOF, LIF, and LRF. The proposed strategy, which we refer to as EMIS (*Extended Meaningful Impact Strategy*), guarantees the detection of these seven types of faults. Although MAX-A also guarantees the detection of all these types of faults, EMIS achieve more efficiency than MAX-A by selecting test cases from each $UTP_i(S)$ and $NFP_{i,j}(S)$ appropriately. EMIS selects points based on the following two steps.

In the first step, points are selected from each $UTP_i(S)$ such that as many values for variables in $V - V_i$ as possible are covered while minimizing the number of selected points.

In the second step, points are selected from each $NFP_{i,j}(S)$ as follows. When there are variables in $V - V_i$ that do not take 0 (1) in the points selected from $UTP_i(S)$ in the first step, new points are selected from $NFP_{i,j}(S)$ such that as many of these variables as possible take 1 (0) while minimizing the number of selected points. If no point can be selected this way, one arbitrary point is selected.

We illustrate the strategy by using $S_1 = \bar{a}bd + \bar{c}d + e$ as an example specification. The test suite for the specification constructed by EMIS is shown in Table 1. Focusing on $UTP_3(S_1)$, we explain the first step. Table 2 lists the points in $UTP_3(S_1)$. In this case, the points '1' and '31' are selected, since all four variables in $V - V_3 (= \{a, b, c, d\})$ take both 1 and 0 when these two points are selected.

Focusing on LRF, which stands for *Literal Reference Fault*, we explain how our strategy ensures fault detection. An LRF on p_3 is a fault such that e is replaced by a literal $x \in \{a, \bar{a}, b, \bar{b}, c, \bar{c}, d, \bar{d}\}$. To detect such faults, points in which x is 0 need to be selected from $UTP_3(S_1)$, since the faulty term of p_3 evaluates to 0 for such points. As mentioned above, all four variables in $V - V_3$ take both 1 and 0 in the two points '1' and '31'. Therefore any faulty term of p_3 with a LRF evaluates to 0 for either '1' or '31'.

We next explain the second step by considering points selection from $NFP_{1,3}(S_1)$. The point '22' is selected from $UTP_1(S_1)$ in the first step. Since in the point c and e do not take 0 and 1, respectively, new points need to be selected from $NFP_{1,3}(S_1)$ such that c and e take 1 and 0, respectively. Therefore the point '20' is selected from $NFP_{1,3}(S_1)$.

For p_1 , some LRFs can not be detected by the points '22' selected in the first step. Concretely, although LRFs such that a is replaced by a literal $x \in \{\bar{c}, e\}$ are detected by the point '22', the rest of LRFs such that a is replaced by a literal $x \in \{c, \bar{e}\}$ are not detected. To detect these LRFs, new points in which x is 1 need to be selected from $NFP_{1,3}(S_1)$, since these faulty term of p_1 evaluate to 1 for such points. c and e in the point '20' take 1 and 0, respectively. Therefore, these LRFs are detected. Although we omit the proof, any faults of the seven types can be detected by EMIS.

We used *mutation analysis* [2] to demonstrate our claim. In this analysis, faulty implementations, called *mutants*, are generated first. A test suite constructed is then evaluated for each specification with *mutation score*, which is the percentage of the number of mutants detected to the total num-

Table 1. A test suite generated by EMIS for $S_1 = \bar{a}bd + \bar{c}d + e$.

#Tests	Point number	a	b	c	d	e	From
1	22	1	0	1	1	0	$UTP_1(S_1)$
2	2	0	0	0	1	0	$UTP_2(S_1)$
3	26	1	1	0	1	0	$UTP_2(S_1)$
4	1	0	0	0	0	1	$UTP_3(S_1)$
5	31	1	1	1	1	1	$UTP_3(S_1)$
6	6	0	0	1	1	0	$NFP_{1,1}(S_1)$
7	30	1	1	1	1	0	$NFP_{1,2}(S_1)$
8	20	1	0	1	0	0	$NFP_{1,3}(S_1)$
9	14	0	1	1	1	0	$NFP_{2,1}(S_1)$
10	24	1	1	0	0	0	$NFP_{2,2}(S_1)$
11	8	0	1	0	0	0	$NFP_{3,1}(S_1)$

Table 2. $UTP_3(S_1)$ of $S_1 = \bar{a}bd + \bar{c}d + e$.

Point number	a	b	c	d	e
1	0	0	0	0	1
5	0	0	1	0	1
7	0	0	1	1	1
9	0	1	0	0	1
13	0	1	1	0	1
15	0	1	1	1	1
17	1	0	0	0	1
21	1	0	1	0	1
25	1	1	0	0	1
29	1	1	1	0	1
31	1	1	1	1	1

ber of mutants. That is, Mutation score = (Number of mutants) / (Number of all mutants) \times 100.

By adopting the seven fault models mentioned earlier, we created 51 mutants for S_1 . The mutation scores obtained by BMIS, MAX-A, and EMIS were 86.3%, 100.0%, and 100.0%, respectively, while the sizes of test suites generated by BMIS, MAX-A, and EMIS were 10, 26, and 11. Although both EMIS and MAX-A guarantee the detection of all the types of faults, the result shows that EMIS is more effective than MAX-A.

References

- [1] T. Y. Chen and M. F. Lau, "Two test data selection strategies towards testing of boolean specifications," *Proc. International Computer Software and Applications Conference (COMPSAC'97)*, pp.608-611, 1997.
- [2] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *IEEE Computer Magazine*, Vol.11, 1978.
- [3] K. A. Foster, "Sensitive test data for logic expressions," *ACM SIGSOFT Software Engineering Notes*, Vol.9, No.2, pp.120-125, 1984.
- [4] D. R. Kuhn, "Fault classes and error detection capability of specification-based testing," *ACM Trans. Software Engineering and Methodology*, Vol.8, No.4, pp.411-424, 1999.
- [5] E. Weyuker, T. Goradia, and A. Singh, "Automatically generating test data from a boolean specification," *IEEE Trans. Software Engineering*, Vol.20, No.5, pp.353-363, 1994.