Towards Selecting Test Data
Using Topological Structure of Boolean Expressions

Lian Yu, Wei-Tek Tsai¹, Wei Zhao², Jun Zhu², Qianxing Wang³
School of Software and Microelectronics, Peking University, P.R. China
¹ Arizona State University, USA; ² IBM China Research Lab
³ Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, P.R. China

Abstract

Boolean expressions can be used in programs and specifications to describe the complex logic decisions in mission-critical, safety-critical, and Web services applications. We define a topological model (T-model) to represent Boolean expressions and characterize the test data. This paper provides proofs of relevant T-model properties, employs the combinatorial design approach, and proposes a family of strategies and techniques to detect a variety of faults associated with Boolean expressions. We compare our strategies with MC/DC, MUMCUT, MANY-A, MANY-B, MAX-A and MAX-B, and conclude that T-model based approach detects more types of faults than MC/DC, MUMCUT, MANY-A and MAX-A, and detects the same types but more instances of faults than MANY-B and MAX-B with much smaller test data set.

1. Introduction

Boolean expressions can be used in programs and specifications to describe the complex logic decisions in many mission-critical and safety-critical applications [1-3], and Web services applications [4]. Researchers have proposed several test data selection criteria or testing strategies based on Boolean expressions in the past two decades.

MC/DC [1] guarantees to detect Expression Negation Fault and Term Negation Fault. The research by Chen et al. [5] shows that MC/DC test sets are effective, but they may still miss some faults that can almost always be detected by test set satisfying the MUMCUT criterion.

Weyuker et al. [2] classify test data space of Boolean expressions into Unique True Points (UTP), Near False Points (NFP), Overlapping True Points (OTP) and Remaining False Points (RFP); and present a family of test cases selection strategies, such as MANY-A, MANY-B, MAX-A and MAX-B, to detect more fault types than MC/DC.

In their paper, Chen et al. declare that MUMCUT has been proved to guarantee the detection of Literal Insertion Fault and Literal Reference Fault in logical decisions that Weyuker’s family strategies fail to discover [5].

Our investigations show that MUMCUT cannot guarantee to detect faults that only can be detected by test data from the following sets:

- \{UTP\} \{MUTP\}: The difference between the set of UTP and the set of MUTP, where MUTP represents the test points that are selected by MUTP strategy. For example, Expression Insertion Fault (EIF) shows this type of faults.
- \{OTP\}: The set of OTP. For example, Double Expression Insertion Fault (DEIF) demonstrates this type of faults.
- \{RFP\}: The set of RFP. For example, Term Insertion Fault (TIF) explains this type of fault.

As Weyuker’s MAX-A selects all \{UTP\}, MAX-A can detect faults pertinent to \{UTP\} \{MUTP\}, but with too many test data as our experiments show. On the other hand, Weyuker’s MAX-B selects the number of (log₂\{OTP\} + log₂\{RFP\}) test data from \{OTP\} and \{RFP\}, respectively, but cannot guarantee to detect the two types of faults pertinent to \{OTP\} and \{RFP\}, yet with too many test data.

To inspect systematically logical faults, this paper explores the topological structure of Boolean expressions and creates a topological model, called T-model, for Boolean expressions. We define a family of strategies to select test data based on the T-model. Our analysis and experiment show that T-model based test strategies not only can guarantee to detect more types of faults than MC/DC, MUMCUT, MANY-A and MAX-A, but also detect more faults and generate much less test data than MANY-B and MAX-B.

The rest of the paper is organized as follows: Section 2 presents the basic concepts and fault types. Section 3 constructs the topological model, T-model, for a Boolean expression, and introduces T-model properties. Section 4 investigates topological characteristics of faults based on T-model. Section 5 presents algorithms and strategies to select test data to detect a variety of faults associated with Boolean expressions. Section 6 performs experiments to validate the proposed algorithms and strategies.
Section 7 surveys related work. Finally, Section 8 concludes the paper and sketches the future work.

2. Notations and Related Work

This section mainly presents notations that are used in the paper.

2.1. Notations and Definitions

The following notations are used throughout the paper. “+”, “·”, and “¬” represent the “or”, “and”, and “not” Boolean operations, respectively. Usually, the “·” is omitted. We use 1 and 0 to denote “true” and “false” correspondingly.

A Boolean expression \( S \) with \( n \) unique variables can be expressed as sum-of-products form or product-of-sums form, known as disjunctive normal form (DNF) and conjunctive normal form (CNF), respectively [2]. If a product-term contains every variable of \( S \), either in complemented or un-complemented form, we call it min-term. A literal is an occurrence of a variable in the Boolean expression, whether it is negative or positive. \( S \) is said to be irreducible disjunctive normal form (IDNF) if \( S \) is DNF, and nor its terms or any of their literals may be omitted from \( S \) without changing \( S \) [2].

A point \( t \) is said to be a true point (false point) of \( S \) if \( S(t) \) evaluates to 1 (0). All true/false points set of \( S \) is represented as \( TP(S)/FP(S) \). For each term \( T_i \) in \( S \), \( t \) is named to be a true point of \( T_i \), if \( T_i(t) \) evaluates to 1. If \( t \) only makes term \( T_i \) in \( S \) true, and other terms false, we call \( t \) as a unique true point (UTP) of term \( T_i \). The sets of all true points and unique true points of \( T_i \) are denoted by \( TP(T_i) \) and \( UTP(T_i) \), respectively. Let \( UTP(S) \) be all unique true points of \( S \), the union of all \( UTP(T_i) \). The set of all overlapping true points (OTP) of \( S \), which are true points but not unique true points, is given by \( OTP(S) = TP(S)/UTP(S) \).

Suppose \( T_{ij} \) is denoted as the term derived from \( T_i \) by negating its \( j \)th literal \( x_{ij} \). A point \( t \) is called as a near false point (NFP) for the \( j \)th literal \( x_{ij} \) of the \( i \)th term \( T_i \) in \( S \) if \( 1 \) \( T_{ij} \) \( (t) = 1 \); and \( 2 \) \( S(t) = 0 \). The set of all near false points for \( j \)th literal \( x_{ij} \) of \( i \)th term \( T_i \) is represented as \( NFP(T_{ij}) \), for term \( T_i \) as \( NFP(T_i) \) and for all terms in \( S \) as \( NFP(S) \). The difference between \( FP(S) \) and \( NFP(S) \) is called remaining false points, \( RFP(S) \).

2.2. Fault Types

We use \( S \) to denote the original Boolean expression. The ten types of faults are briefly described as follows [5]:

1) Expression negation fault (ENF) designates that the entire or sub of \( S \) is replaced by its negation.
2) Term negation fault (TNF) designates that one term of \( S \) is replaced by its negation. It is a special case of ENF, i.e., only one term of \( S \) is negated.
3) Term omission fault (TOF) means that one term of \( S \) is omitted.
4) Operator reference fault (ORF) refers to either a disjunctive ORF (DORF) which replaces an OR operator by AND operator, or a conjunctive ORF (CORF) which replaces an AND by OR.
5) Literal insertion fault (LIF) means that a literal is inserted into one term of \( S \). It is also known as variable insertion fault.
6) Literal negation fault (LNF) is that a literal is replaced by its negation. It is also known as variable negation fault.
7) Literal omission fault (LOF) means that a literal is omitted. It is also known as variable omission fault.
8) Literal reference fault (LRF) means that a literal is replaced by another literal. It is also known as variable reference fault.
9) Stuck-at fault (SAF) refers to either a stuck-at-0 fault (SAF[0]) or a stuck-at-1 fault (SAF[1]), which causes the value of a literal to be stuck at 0 or 1, respectively.
10) Associative shift fault (ASF) is that the associativity of terms is changed once.

It has been proved that test data selected according to MUMCUT strategy can detect the first nine types of faults and with high probability to detect the last one [6]. Our research demonstrates that there are more faults that MUMCUT cannot guarantee to detect, for example, the following two types of faults [7]:

11) Term insertion fault (TIF) has two forms. One is that to insert a term to a term of \( S \) using “And” operator; the other is to insert a term to \( S \) using “Or” operator, and they are symbolized as TIF(·) and TIF(+), respectively.
12) Expression insertion fault (EIF), akin to term insertion fault, also has two forms. One is that to insert an expression to a term of \( S \) using “And” operator. The other is to insert an expression to \( S \) using “Or” operator. We use EIF(·) and EIF(+) to denote them.

This paper will provide strategies and techniques to detect those kinds of faults.

3. Constructing T-models for Boolean Expressions

This section describes a topological structure of Boolean expressions, called T-model, based on Karnaugh map (a.k.a. K-map) [8]. Moreover, we define the Hamming distances for expression terms
and cells to characterize the topological properties of the T-model.

3.1. Topological Structure of a Boolean Expression

This paper uses K-map to explore the topological structure, T-Model, of Boolean expression. The T-Model is similar to K-map, and Boolean variables are represented along axes, and arranged in such a way that only one value changes in between squares or cells [9].

Given a Boolean expression \( S = ac\overline{d} + ab + b\overline{c}\overline{d} \), letting \( x \) axis represent Boolean variables \( a \) and \( b \), and \( y \) axis represent variables \( c \) and \( d \), T-Model of \( S \) is shown in Figure 1. Accordingly, term \( ac\overline{d} \) is transferred to cells \( C_1=\{1110, 1010\} \) on top right of T-model; term \( ab \) to cells \( C_2=\{1000, 1001, 1011, 1010\} \) on the most right of T-model; and term \( b\overline{c}\overline{d} \) to cells \( C_3=\{0100, 1100\} \). Note that \( S = C_1 \cup C_2 \cup C_3 \).

These cells marked as “1” on the map indicate that they have \( S \) evaluated to “true”, and the rest of other cells marked as “0” indicate that they have \( S \) evaluated to “false”. Each cell in T-Model corresponds to a min-term, such as cell \( \{1110\} \) in T-Model matches up to min-term \( abc\overline{d} \) of \( S \).

![Fig. 1. K-map for \( S = ac\overline{d} + ab + b\overline{c}\overline{d} \)](Image)

The arrangement of Boolean variables and the start point does not affect the topological structure of Boolean expressions in terms of min-terms and their relative positions. The T-Model of Figure 2(a) is obtained from Figure 1 by moving the start point from the lower left corner to the upper left corner, and keeping \( ab \) and \( cd \) orientation the same. The min-terms of \( S \) do not change and the relative positions of terms in \( S \) do not change either, but flipping the “1” cells 180 degrees around horizontal axis.

The T-Model of Figure 2(b) is obtained from Figure 1 by exchanging \( ab \) and \( cd \) orientations. The min-terms of \( S \) do not change and the relative positions of terms in \( S \) do not change either, but turning diagonally the “1” cells 180 degrees around.

The T-Model of Figure 2(c) is obtained by exchanging orientations of variables \( b \) and \( c \). It seems that the shape of T-Model in Figure 2(c) looks very different from that of Figure 1. Cells of term \( ac\overline{d} \) and cells of \( b\overline{c}\overline{d} \) are split apart respectively, and four long rectangles of term \( ab \) in Figure 1 are transformed into four squares in Figure 2(c). Note that the grid on a T-Model is toroidally connected, which means that the rectangular groups can wrap around edges, so cell \( \{1110\} \) is adjacent to cell \( \{1100\} \) and they shape term \( ac\overline{d} \). The min-terms of \( S \) do not change and the relative positions of terms in \( S \) do not change either, but flowing the “1” cells around.

3.2. Hamming Distance in T-Model

We define an attribute on the topological structure of Boolean expression to help describe our strategy on test data selection: Hamming distance (HD), which is named after R.W. Hamming [10]. It is usually defined as the number of different bits between two binary strings. In T-Model, the meaning of HD is defined as follows.

**Definition 1**: With regard to Boolean expression \( S \), suppose that a cell, \( c \), needs at least \( p \) (\( p \geq 1 \)) steps to reach the cell, \( c' \), and \( S(c) \oplus S(c') = 1 \). If \( S(c) = 1 \), the HD of \( c \) is \( p \), otherwise the HD of \( c \) is \( -p \).

According the definition of HD, we can derive the follow conclusion. In T-Model, \( \{TP\} \) of a Boolean expression corresponds to cells, \( C_{\text{HD}=1} \), with \( HD \geq 1 \); and \( \{FP\} \) corresponds to cells, \( C_{\text{HD}=-1} \) with \( HD \leq -1 \).

![Fig. 3. HDs of a Boolean expression](Image)

To simplify drawing a T-Model of a Boolean expression, hereafter, we use shadowed cells to represent “1” cells, and empty cells to “0” cells. In Figure 3, HDs of cells \( \{0101, 0111, 0110, 1101, 1110, 1001, 1011\} \) are all equal to \( 1 \), \( HD \) of \( \{1111\} \) is equal to \( 2 \), and rest are equal to \(-1 \). In T-Model, changing a bit in min-term is equivalent to moving a cell one step horizontally or vertically to a neighboring cell. For example, changing a bit of min-term \( \{0110\} \) to \( \{0010\} \)
means moving a step horizontally from cell \{0110\} to cell \{0010\} in the T-Model of Boolean expression \(S\).

In T-Model, cells with \(HD \geq 1\) can be used as test data for positive testing where cells with \(HD = 1\) form the positive boundary of a Boolean expression, while cells with \(HD \leq -1\) can be used as test data for negative testing, where cells with \(HD = -1\) form the negative boundary of a Boolean expression.

Hereafter, we will use T-Model to describe our strategies to select test data to detect faults associated with Boolean expression. When a Boolean expression has more than 6 variables, it is not recommended to draw a diagram visually to represent a T-model of an expression. Instead, we provide algorithms to automatically search test data in T-model.

3.3. Points Partition on T-model

Test data selection strategies of BMIS, MANY-A, MAX-A, and MUMCUT [2] are exclusively based on UTP and NFP; MC/DC [1] based on TPs and NFP; while MANY-B and MAX-B randomly select certain number of test data from OTP and RFP [2] in addition to UTP, and NFP.

**Theorem 1.** In a T-model, \{NFP\} of term \(T_i\) corresponds to cells with \(HD = -1\); and \{RFP\} corresponds to cells, \(C_{HD=2}\) with \(HD \leq -2\).

**Proof.** Let \(T_i\) be a term of Boolean expression \(S\), \(T_i = x_{i,1} \cdot x_{i,2} \cdot \ldots \cdot x_{i,j} \cdot \ldots \cdot x_{i,k}\). According to the definition of NFP, \{NFP\} of term \(T_i\) satisfies \(T_i = x_{i,1} \cdot x_{i,2} \cdot \ldots \cdot x_{i,j} \cdot \ldots \cdot x_{i,k}\) where \(S(T_i) = 1\), \(S(T_{i,j}) = 0\), and the only difference between \(x_{i,j}\) and \(x_{i,j}\), i.e., to move from a cell of \(T_i\) to a cell of \(T_{i,j}\), only one step is needed.

That exactly matches the definition of \(HD=-1\), therefore set \{NFP\} of term \(T_i\) corresponds to cells with \(HD = -1\). For \{RFP\}, at least two variables are flipped regarding to term \(T_i\), which corresponds to at least two step moves on T-model from positive cells to negative cells. That is cells with \(HD \leq -2\).

On the other hand, both \{UTP\} and \{OTP\} may have cells with \(HD \geq 2\), i.e., two sets may have cells within the positive boundary of a Boolean expression. In addition, the two sets depend on the form of Boolean expressions. If Boolean expressions are disjoint disjunctive normal form (DDNF), \{OTP\} is always empty; if Boolean expressions are in IDNF, \{OTP\} may not be empty.

4. More Boolean Expression Faults

In addition to the ten types of faults that can be detected by MUMCUT as described in Section 2.2, there are other types of faults that cannot be detected by MUMCUT at all. The hierarchy of those faults is discussed in paper [7]. A cell can detect a fault if the cell makes a Boolean expression and its mutant with a fault evaluated to different outcomes.

4.1. More Faults Associated with UTP

EIF(·) is a type of faults related to UTP, which cannot be found by MUMCUT. For example, \(S = ab + cd\) becomes \(S_1 = ab(c + d) + cd\) when the new expression \((c + d)\) is inserted into the term \(ab\) as shown in Figure 4.

MUTP strategy [5] selects test data from \{UTP\} of the term \(ab\) such that not-appearing variables \(c\) and \(d\), occur at “0” and “1” as much as possible. Test set \{1101, 1110\} meets MUTP strategy but cannot guarantee to detect the fault of EIF as the test set always makes \(S_i\) evaluated to be true, and does not differentiate \(S\) from \(S_1\).

\[
S = ab + cd \quad S_1 = ab(c + d) + cd
\]

Fig. 4. An EIF fault in T-model

4.2. More Faults Associated with OTP

There are some faults that can only be detected by test data from \{OTP\}. These faults cannot be discovered by test data based on MUMCUT since MUMCUT does not select those test data at all.

• Double expression insertion fault (DEIF) means that two new expressions are inserted into two different terms of a Boolean expression. For example, \(S\) becomes \(S_2 = ab(\overline{c + d}) + cd(\overline{c + \overline{d}})\) where the expression \((\overline{c + d})\) is inserted to the term \(ab\), and the expression \((\overline{c + \overline{d}})\) is inserted to the term \(cd\) at the same time as shown in Figure 5.

\[
S = ab + cd \quad S_2 = ab(\overline{c + d}) + cd(\overline{c + \overline{d}})
\]

Fig. 5. T-model for \(S_2 = ab(\overline{c + d}) + cd(\overline{c + \overline{d}})\)

4.3. More Faults beyond NFP

There are some faults that must be detected by test data with \(HD \leq -2\). As MUMCUT strategy only selects
negative test data from those with \( HD = -1 \), it cannot reveal those faults.

For example, in Boolean expression \( S = abc + abd \), the first term \( abc \) is written as \( c(ab + \overline{a}b) \), and \( S \) becomes \( S_t = c(ab + \overline{a}b) + abd \) as shown in Figure 6(a); or there is an extra term \( \overline{a}b \overline{c}c \) added to \( S \), resulting in \( S_t = abc + \overline{a}b \overline{c}c + abd \) as shown in Figure 6(b). The faults are a kind of TIF(+) with \( HD \leq -2 \).

The next section will describe an algorithm to search the cells with \( HD \leq -2 \), and then present a combinational design based approach to finding these kinds of faults described in sections 4.2 through 4.4.

5. T-model based Test Data Selection Strategies

Let \( n \) be the number of Boolean variables in a Boolean expression. \( HDs \) of the Boolean expression vary from \( n \) to \( -n \). We have developed a family of algorithms to calculate overlapping true points, unique true points, near false points, and generally, the test set with \( -w \leq HD \leq w \). Due to the limitation of page numbers, we only present an algorithm to search cells with \( HD = -w \), where \( 2s w \leq n \).

5.1. Finding Cells with \( HD = -w \)

**Definition 2:** Distance of a cell \( c \) to a term \( T_i \) is \( q \) (\( q \geq 1 \)), if \( c \) needs at least \( q \)-step moves to reach a true point of \( T_i \).

Given a Boolean expression \( S \) in sum-of-product form and a parameter \( HD = -w \), the algorithm will output a set \( C_{HD = -w} \), storing cells or \( \min \)-terms with \( HD = -w \).:

1. Initialize the set, \( C_{HD = -w} = \emptyset \).
2. Pick up a term \( T_j \) of \( S \), such that it has the maximum number \( \xi \) of appearing variables. Check if the number \( \xi \) is less than \( w \), return \( C_{HD = -w} = \emptyset \) with empty, as in this case, there are no cells with \( HD = -w \) for expression \( S \).
3. Scan a term \( T_j \) in \( S \) and proceed as follows:
   1) Find out all the cells whose distance to term \( T_j \) is \( -w \). Eliminate the cells that make other terms of \( S \) to be true.
   2) Check if each of the cells is nearer to the other terms of \( S \) than \( T_i \). If true, delete it.
   3) Add the rest cells into the set, \( C_{HD = -w} \).
4. Iterate steps 1) through 3) for each of terms in \( S \).

Take the Boolean expression \( S = ab + cd \) as an example to search the cells with \( HD = -3 \) or \( HD = -2 \). As the number of appearing variables for both terms in \( S \) is less than 3, so \( C_{HD = -3} = \emptyset \). For the term \( ab \), there are 4 cells \{0000, 0001, 0011, 0010\} whose distance to \( ab \) is 2, but \{0011\} is a \( UTP \) of the term \( cd \), while \{0001, 0010\} are two \( NFPs \) of \( cd \). Therefore, the cell \{0000\} is the test data with \( HD = -2 \) for the term \( ab \). Likewise, we can calculate that test data with \( HD = -2 \) for the term \( cd \) is also cell \{0000\}. So \( C_{HD = -2} \) is \{0000\}.

5.2. Combinatorial Design based Selection of Test Data

Pairwise Testing is one combinatorial strategy that achieves the pair-wise coverage, in which every possible pair of the values of any two test input parameters is covered by at least one test case. It has been proved to be a cost effective test case generation method in practice [11]. This paper uses the pair-wise approach to select test data from sets \( \{UTP\} \) and \( \{OTP\} \) of terms, and the cells with \( HD = -w \) (\( 1 \leq w \leq n \)) to reduce the number of test cases while maintaining the same capability of defect detection.

Take the Boolean expression \( S = abd + \overline{a} \overline{c}d + abef \) as an example to select negative test data from cells with \( HD = -2 \), which are indicated with “-2” in the corresponding cells in Figure 7. There are 15 cells with \( HD = -2 \) for the expression, and they can be represented in 4 terms, \( \overline{a}d, \overline{b}c, \overline{c}d \overline{e} \), and \( \overline{e}f \), called mutation terms, as mutants of \( S \) may be generated by adding a mutation term to \( S \). For example, \( S \) becomes \( S_t = (ab + \overline{a}b)d + \overline{a} \overline{c}d + abef \) by adding a mutation term \( \overline{a}d \). The fault of \( S_t \) can be detected by any cell of the set \{000100, 001100, 000101, 001101, 000111, 001111, 000110, 001100\}, which are obtained by enumerating not-appearing variables \( c, e \), and \( f \) of mutation term \( \overline{a}d \).

![Fig. 6. More faults beyond NFP](image)

![Fig. 7. T-model for S=abd+\overline{a}c\overline{d}+abef](image)
The following will explain the strategy to select test data, and illustrate to what extent those test data based on combinatorial design approach can detect faults associated with $HD \leq 2$.

1) Select mutation terms such that they have maximum number of not-appearing variables.

2) Choose test data such that all the possible combinations of not appearing variables should show up; and in the meantime, the appearing but different literals among the terms should satisfy the pair-wise condition as much as possible.

For the above example, 4 test data out of 12 candidates are selected, \{000100, 001111, 001001, 001010\}, which satisfy the pair-wise selection strategy. It is observed that not only can the 4 test data detect the faults that result in $S$ plus one of the mutation terms $d b a$ and $c b a$, but also detect the faults that result in $S$ plus one of mutation terms $d c b a$, $cd b a$, $d c b a$, $e d b a$, $de b a$, $f d b a$, $df b a$, $e c b a$, $ce b a$, $d c b a$, $cf b a$, $e d c a$, and $f d c a$.

6. Experiments

This section compares the test strategies of MC/DC, MUMCUT, MANY-A, MANY-B, MAX-A, MAX-B and T-model in terms of sizes of test data and fault detection capabilities. We use the 20 Boolean expressions obtained from TCAS II [2] as the experiment targets. The Boolean expressions vary in size of Boolean variables from 5 to 14, with the average containing 10 distinct variables.

MUMCUT and Weyuker’s approaches require using IDNF as input format. To make comparison consistent, the 20 Boolean expressions are translated to IDNF used as inputs into each test strategy. Notice that when translated into IDNFs, some variables in Boolean expressions 1, 3, 5, 7, 14, 16, and 18 are not appearing in the expressions. It means those not appearing variables actually do not affect the outcomes of the expressions, and some literature calls them as “don’t care” variables of Boolean expressions [12].

6.1. Comparison of Test Set Sizes

Table 1 shows the sizes of the test sets generated by the different test strategies for the 20 Boolean expressions, and the last row shows the average value of the size for each strategy. On average, MC/DC gets the smaller test set size, while MAX-B gets the maximal size.

MANY-A has to select the number $\log_2|\{UTP\}| + \log_2|\{NFP\}|$ of test data, and when the number of $\{UTP\}$ and $\{NFP\}$ becomes large, $\log_2|\{UTP\}| + \log_2|\{NFP\}|$ goes along with. This is the situation for MANY-A for expressions 3, 11 and 12, where the numbers of $\{UTP\}$ have the numbers of 2,392, 1,122, 11,323 respectively, while MUMCUT just needs to choose possible values of not appearing variables, that is the size of MUMCUT does not necessarily go along with the number of $\{UTP\}$.

On average, T-model has the test data size, greater than MC/DC, MUMCUT, MANY-A and MANY-B, and less than MAX-A and MAX-B. More specially, T-model generates less test data than MAX-B, except for expressions 1, 6, 7, 8, 10, 11 and 12. Note that these expressions have types of faults that MAX-B cannot detect while T-model addresses those types of faults and generates corresponding test data to detect them all.

<table>
<thead>
<tr>
<th></th>
<th>MC/DC</th>
<th>MUMCUT</th>
<th>MANY-A</th>
<th>MANY-B</th>
<th>MAX-A</th>
<th>MAX-B</th>
<th>T-model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>16</td>
<td>10</td>
<td>16</td>
<td>10</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>18</td>
<td>12</td>
<td>18</td>
<td>12</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>26</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>13</td>
<td>34</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>36</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>15</td>
<td>38</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>16</td>
<td>40</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
</tbody>
</table>

The time, $T$, to generate test cases based on T-model approach consists of two parts: $T_\Omega$ for searching test data $\Omega$ that meet the test criteria; and $T_D$ for selecting test data $D$ from $\Omega$ to meet pair-wise criteria. The experiment shows that $T_\Omega$ of the 20 formulae varies from 16 milliseconds to 58 seconds while $T_D$ of the 20 formulae varies from $4.16 \times 10^{-4}$ milliseconds to $1.99 \times 10^{-3}$ milliseconds. Therefore, the portion of time spent on combinatorial design computing is very low, while combinatorial design computing helps dramatically reducing the test size.

6.2. Comparison of Detection Capability

Table 2 shows the detection capability regarding the 10 faults presented in Section 2.2.

<table>
<thead>
<tr>
<th></th>
<th>MC/DC</th>
<th>MUMCUT</th>
<th>MANY-A</th>
<th>MANY-B</th>
<th>MAX-A</th>
<th>MAX-B</th>
<th>T-model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>16</td>
<td>10</td>
<td>16</td>
<td>10</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>18</td>
<td>12</td>
<td>18</td>
<td>12</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>18</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>26</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

The time, $T$, to generate test cases based on T-model approach consists of two parts: $T_\Omega$ for searching test data $\Omega$ that meet the test criteria; and $T_D$ for selecting test data $D$ from $\Omega$ to meet pair-wise criteria. The experiment shows that $T_\Omega$ of the 20 formulae varies from 16 milliseconds to 58 seconds while $T_D$ of the 20 formulae varies from $4.16 \times 10^{-4}$ milliseconds to $1.99 \times 10^{-3}$ milliseconds. Therefore, the portion of time spent on combinatorial design computing is very low, while combinatorial design computing helps dramatically reducing the test size.
For convenience, we generate one fault for each expression, and each one represents one fault of the 10 types (Note, as to SAF type, generate one for SAF[0], and one for SAF[1]). It is observed that MUMCUT, MAX-A, MAX-B and T-model can detect all the 11 faults. Although MANY-B generates more data than MANY-A does, MANY-B does not guarantee detecting more faults than MANY-A regarding the 11 faults.

For example, MANY-B failed to one fault in expression 5, 10 and 18, but MANY-A detects them. However MANY-A does not guarantee to detect the 11 faults either; for example, it failed to detect one fault in expression 7. Excepting expressions 3, 5, 14 and 16, MC/DC failed to detect some of the faults, and has weakest detection capability pertinent to the 11 faults.

The second column of Table 3 shows the faults that can only be detected by test data $\{\text{UTP}\} \setminus \{\text{MUTP}\}$, and the rest of columns demonstrate the capabilities of the seven strategies to detect those faults in terms of the numbers. Note that expressions 6, 8, 9, 10, 16 and 20 have no this type of faults. It is observed that MUMCUT fails to detect all these faults. MC/DC has possibility to select test data from $\{\text{UTP}\}$, and so can detect some of the faults but failed to guarantee to detect them all.

Table 3: Detecting faults related to $\{\text{UTP}\} \setminus \{\text{MUTP}\}$

MAX-A, MAX-B and T-model can guarantee detect them all, but with different strategies. MAX-A and MAX-B select all points from $\{\text{UTP}\}$, therefore, they would not miss any faults associated with $\{\text{UTP}\} \setminus \{\text{MUTP}\}$. T-model selects all possible pairwise combination of not appearing variables from the set $\{\text{UTP}(T_i)\}$ of each term $T_i$, and guarantees to find those faults.

The second column of Table 4 shows faults associated with $\{\text{OTP}\}$. Note that expressions 9 and 20 do not have any such type of faults. MC/DC, MUMCUT, MANY-A and MAX-A do not able to detect faults associated with $\{\text{OTP}\}$, while MAX-B and MAX-B select randomly the number of $\log_2|\{\text{RFP}\}|$, and can detect some of such faults, but do not guarantee to detect them all, while T-model is able to detect them all.

Table 4: Detecting faults related to $\{\text{OTP}\}$

It is interesting to notice that criterion MAX-B subsumes criterion MANY-B, but the subsumption of criteria does not mean the subsumption of fault detection capability. For example, MANY-B detects more faults associated with $\{\text{HD}=-2\}$ than MAX-B for expressions 15 and 19.

Similarly, Table 5 shows faults associated with $\{\text{HD}=-3\}$. Note that expressions 2-5, 9, 12-16 and 20 have no such type of faults. Again, MANY-B and MAX-B can detect some of such faults, but cannot detect them all, and the detection capability of MAX-B does not preponderate over that of MANY-B (e.g., expressions 1, 7, 8, 10, 17 and 18). T-model guarantees to detect them all.

Table 5: Detecting faults related to $\{\text{HD}=-3\}$

It is interesting to notice that criterion MAX-B subsumes criterion MANY-B, but the subsumption of criteria does not mean the subsumption of fault detection capability. For example, MANY-B detects more faults associated with $\{\text{HD}=-2\}$ than MAX-B for expressions 15 and 19.

Similarly, Table 6 shows faults associated with $\{\text{HD}=-3\}$. Note that expressions 2-5, 9, 12-16 and 20 have no such type of faults. Again, MANY-B and MAX-B can detect some of such faults, but cannot detect them all, and the detection capability of MAX-B does not preponderate over that of MANY-B (e.g., expressions 1, 7, 8, 10, 17 and 18). T-model guarantees to detect them all.

Table 6: Detecting faults related to $\{\text{HD}=-3\}$
As shown in Table 1, T-model generates more test data than MAX-B for expressions 1, 6, 7, 8, 10, 11 and 19. How will the test results change if MAX-B has the same sizes of test data as T-model does for the above expressions? Table 7 shows that even MAX-B has the same size of test data as T-model does, for the increased rate of detected faults associated with $HD_{\leq -2}$ and $HD_{\leq -3}$ are 14.18% and 13.18%, respectively.

Table 7: Increasing MAX-B Strategy's test sizes

<table>
<thead>
<tr>
<th>Id</th>
<th>Total faults</th>
<th>Before</th>
<th>Other</th>
<th>Increased rate</th>
<th>Total faults</th>
<th>Before</th>
<th>Other</th>
<th>Increased rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>288</td>
<td>4</td>
<td>5</td>
<td>37.90%</td>
<td>16</td>
<td>5</td>
<td>5</td>
<td>8.57%</td>
</tr>
<tr>
<td>6</td>
<td>288</td>
<td>2</td>
<td>8</td>
<td>27.43%</td>
<td>408</td>
<td>0</td>
<td>8</td>
<td>13.36%</td>
</tr>
<tr>
<td>7</td>
<td>162</td>
<td>6</td>
<td>8</td>
<td>8.57%</td>
<td>162</td>
<td>15</td>
<td>5</td>
<td>2.23%</td>
</tr>
<tr>
<td>8</td>
<td>288</td>
<td>2</td>
<td>17</td>
<td>5.80%</td>
<td>288</td>
<td>0</td>
<td>15</td>
<td>14.05%</td>
</tr>
<tr>
<td>10</td>
<td>848</td>
<td>11</td>
<td>473</td>
<td>65.00%</td>
<td>848</td>
<td>24</td>
<td>66</td>
<td>26.51%</td>
</tr>
<tr>
<td>11</td>
<td>689</td>
<td>5</td>
<td>32</td>
<td>3.13%</td>
<td>689</td>
<td>0</td>
<td>40</td>
<td>2.66%</td>
</tr>
<tr>
<td>19</td>
<td>162</td>
<td>19</td>
<td>21</td>
<td>1.08%</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>0.02%</td>
</tr>
</tbody>
</table>

In addition, we choose expressions #12 and #17 to examine the detection capability using the pair-wise strategy. We pick mutation terms associated with $HD_{\leq -2}$ as examples. Table 8 shows the numbers of all possible mutation terms with not-appearing variables varying from 8 to zero for expressions #12 and 6 to zero for expression #17. Each mutation term corresponds to a mutant of the Boolean expression.

Table 8: Pair-wise application impact

<table>
<thead>
<tr>
<th>N</th>
<th>Boolean Expression #12</th>
<th>Boolean Expression #17</th>
<th>A</th>
<th>All</th>
<th>Detected</th>
<th>Percentage</th>
<th>All</th>
<th>Detected</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>15</td>
<td>15</td>
<td>100.00%</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>194</td>
<td>191</td>
<td>98.45%</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1001</td>
<td>928</td>
<td>92.71%</td>
<td>6</td>
<td>6</td>
<td>100.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2831</td>
<td>2450</td>
<td>86.56%</td>
<td>43</td>
<td>36</td>
<td>83.72%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5923</td>
<td>3798</td>
<td>76.93%</td>
<td>164</td>
<td>101</td>
<td>61.59%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5327</td>
<td>3451</td>
<td>62.44%</td>
<td>374</td>
<td>158</td>
<td>42.25%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3213</td>
<td>1780</td>
<td>45.72%</td>
<td>489</td>
<td>126</td>
<td>25.77%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1607</td>
<td>991</td>
<td>30.55%</td>
<td>333</td>
<td>46</td>
<td>13.81%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>293</td>
<td>49</td>
<td>16.72%</td>
<td>91</td>
<td>6</td>
<td>6.50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We generate test data only for the mutation terms with the maximum number (8 and 6, in this case) of the not-appearing variables, using the pair-wise approach as described in Section 4.2. It is observed that the test data can detect not only the faults of targeted mutation terms, but also the faults caused by other mutation terms, with detection rates from 98.45% to 16.72% for expression #12, and from 83.72% to 6.59% for expression #17. The less the number of not-appearing variables, the higher the fault density is, and it needs more test data to detect. This explains the reason that the pair-wise approach reduces the efforts to generate more test data, and has the higher detection capability.

6.3. Statistical Analysis

It is observed from tables 4 through 7 that T-model has higher detection rate (the number of detected faults/total number of faults) than MAX-B. From now on, we will refer to this hypothesis as H, and analyze if the hypothesis is supported by the experiment results. Let the null hypothesis $H_0$ be that “T-model and MAX-B have the same detection rate”.

As the differences of T-model and MAX-B fail to fit a theoretical probability distribution, we use the permutation test, a non-parametric test with no distributional assumptions, to evaluate our hypothesis. A permutation test is a type of statistical significance test in which a reference distribution is obtained by calculating all possible values of the test statistic under the rearrangements of the labels on the observed data points [13].

We apply permutation test to determine the observed differences between the sample means are large enough to reject the null hypothesis $H_0$. The test proceeds as follows.

First, the absolute value of the difference in means of detection rate between the two samples is calculated. This is the observed value of the test statistic, $T = \text{abs}(\mu_{T-model} - \mu_{MAX-B}) = 0.997647 - 0.564008 = 0.433639$.

We then calculate the number of ways of grouping the paired values into 2 sets, with a sample size of 20 paired values, $\text{Number of Permutations} = 2^{20} = 1048576$. Let $Count$ be the number of permutations with absolute difference in means greater than or equal to the observed value, $T$. For our sample data, $Count = 32$. The two-sided $P$-value for the test is calculated as:

$$P\text{-value} = \frac{Count}{\text{Number of Permutations}} = \frac{32}{1048576} = 0.00003$$

Since our $P$-value ($0.00003$) is much less than the significance level $\alpha$ ($0.05$), the null hypothesis $H_0$ would be rejected. Furthermore, as the difference of sample means ($0.433639$) is larger than $0$, we can say the hypothesis H that the detection rate of T-model is higher than the detection rate of MAX-B is statistically significant.

6.4. Discussion on Experiments

The experiments show T-model approach produces good balanced results in terms detection capability and test size.

Fault-based testing strategies are effective to detect the faults that are assumed to frequently occur. It generates test cases to discover those specific defects. However, if it is not the case, fault-based testing is not better than random testing. That is if some faults mentioned in this paper, including the eleven types of faults, $\{OTP\}, \{UTP\}, \{MUTP\}$ related faults and $HD_{\leq -2}$ related faults, would never happen, generating those test cases would never work to improve the quality of software. The experiments report promising results based on fault-based approach. To really incubate the achievements, we need to predicate the types of faults based on historical or
relevant data. This will be the direction of our future research.

7. Related Work

The study on the generation of test cases from Boolean expressions can be dated back to more than two decades ago. Yu and Lau [6] have presented a comprehensive survey on related work by Foster [14], Tai and Su [15], Chilenski and Miller [16], Dupuy and Leveson [17], Jones and Harrold [18], Weyuker et al. [2], Kobayashi et al. [16], Chen and Lau [5], and Kuhn [19]. In the following, we include additional related work that were published afterwards or closely related to this paper.

Lau and Yu analyze the double faults detection ability of the MUMCUT test strategy, and claim that although MUMCUT guarantees to find the above faults in IDNF Boolean expressions, but cannot guarantee to detect the double faults combined with the single faults [20-22]. Lau divides the issue into three parts: double fault combining with two single faults related to a literal, double faults making up of two single faults related to a term, and two faults constituting with one literal and one term.

In paper [21], Lau et al. analyze the combinations of the five term-related faults (ENF, TNF, TOF, DORF, CORF) with/without ordering, where they decompose ORF into CORF and DORF, and they examine the fault coupling effects. They find out that “any test case selection strategy which subsumes the BASIC meaningful impact strategy can detect all double faults” related to terms. However, these double faults related to terms are still related to UTP and NFP with $HD\geq 1$ and $HD=1$, and never incur the faults with $HD<-1$.

In paper [22], the authors studied all the double faults related to the four single literal faults, LNF, LIF, LOF, and LRF. They derive 19 double fault combinations, and find out that 6 of them can not be detected by MUMCUT. The 6 double faults are the combinations of LOF and LRF, LIF and LIF, LIF and LRF, LRF, and LRF. In order to satisfy the detection condition of these 6 double-faults, they propose six test case selection strategies to supplement the MUMCUT, including PMNFP (Pairwise Multiple Near False Points), SMFP (Supplementary Multiple False Point), SPMFP (Supplementary Pairwise Multiple False Point), PMUTP (Pairwise Multiple Unique True Point), SMUTP (Supplementary Multiple Unique True Points), and SMOTP (Supplementary Multiple Overlapping True Point). The research on cost-effectiveness of the new six test strategies is ongoing. As these double faults related to the literal must be detected by test data related to the OTP, that is why MUMCUT can never discover them. T-model based strategy generates test cases from $\{HD \geq 1\}$ and $\{HD=1\}$, therefore can identify those faults.

In paper [20], Lau studied all the double faults combination cross term and literal with/without ordering. By comparing and analyzing these combinations, there are only 38 unique double faults on term and literal at last. They find out only 2 faults of them need SMOTP supplementing MUMCUT. In other words, these double faults are faults related to $\{HD=-1\}$ and $\{HD\geq 1\}$, not related to $\{HD<-1\}$ as well. T-model can generate test cases by selecting right $UTP$, $OTP$ and $NFP$ from $\{HD=-1\}$ and $\{HD\geq 1\}$ to detect those faults.

Their studies on the detection conditions of faults in Boolean expressions show that MUMCUT can also find all double faults related to terms, however, it does not guarantee to detect the double faults composed by two single literal faults or one literal one term. To detect all double faults, Lau and Yu propose several new test case selection strategies as the MUMCUT strategy’s supplements.

Kaminski et al. [23] defines some faults related to $TIF(\cdot)$ and also extends the hierarchy of faults. But $TIF(\cdot)$ is just one type of faults related to $\{RFP\}$. In addition, $EIF(\cdot)$ is also a type of faults related to $\{RFP\}$.

Their paper does not specifically provide algorithms to generate test cases to detect those faults.

Tsai et al. have defined Hamming distances onto a K-map, and based on the model, which they call “Swiss Cheese”, they generate test cases for Web Services testing [4]. This paper defines a similar model, called T-model. Compared with “Swiss Cheese” model, this paper has the following new contributions:

1) Address the theoretical aspects of the approach, explore formally the topological properties, and provide the proofs of the relevant theorems.
2) Introduce a combinatorial design approach to reduce the number of test data and maintain the same detection capability.
3) Investigate the faults that cannot be detected by MUMCUT, and design the test strategies based on T-model to detect those faults.
4) Perform empirical studies to compare the detection capabilities with MC/DC, MUMCUT, MANY-A, MANY-B, MAX-A and MAX-B.

8. Conclusion and Future Work

This paper investigates the faults associated with Boolean expressions and finds out that most of existing researches either could not detect some types of faults or generates too many test data. We develop a topological model, called T-model, to portray test data in terms of Hamming Distances. Based on the T-model, we provide a family of strategies to detect various
faults associated with Boolean expressions. To reduce the test data size and maintain the high detection capability, this paper introduces a combinatorial design approach, pair-wise strategy. To evaluate our proposed approach, we compare with MC/DC, MUMCUT, MANY-A, MANY-B, MAX-A and MAX-B. The experiment results show that T-model based approach detects more types of faults than MC/DC, MUMCUT, MANY-A and MAX-A, and detects the same types but more instances of faults than MANY-B and MAX-B with much smaller test data set. The future work will enhance theoretical analysis on the proposed approaches, provide mathematical proofs, and experiment on more types of applications.

9. Acknowledgement

The work presented in this article is partly sponsored by IBM Shared University Research Grant and by the National High-Tech Research and Development Plan of China (No. 2006AA01Z175). The authors would thank Xiangdong Fan, Guan Wang, Jun Fan, Hongbo Chen, and Bo Hou for working on the empirical study described in this paper.

References