Comparing Several Coverage Criteria for Detecting Faults in Logical Decisions∗

Y. T. Yu†
Department of Computer Science
City University of Hong Kong
Kowloon Tong, HONG KONG
csytyu@cityu.edu.hk

M. F. Lau
School of Information Technology
Swinburne University of Technology
Hawthorn 3122, AUSTRALIA
elau@it.swin.edu.au

Abstract

Many testing coverage criteria, including decision coverage and condition coverage, are well-known to be inadequate for software characterised by complex logical decisions, such as those in safety-critical software. In the past decade, more sophisticated testing criteria have been advocated. In particular, compliance of MC/DC has been mandated in the aviation industry for the approval of airborne software. On the other hand, the MUMCUT criterion has been proved to guarantee the detection of certain faults in logical decisions in irredundant disjunctive normal form. We analyse and empirically evaluate the ability of test sets satisfying these testing criteria in detecting faults in logical decisions. Our results show that MC/DC test sets are effective, but they may still miss some faults that can almost always be detected by test sets satisfying the MUMCUT criterion.

Keywords: Condition coverage, coverage criteria, decision coverage, logical decisions, MC/DC, MUMCUT

1 Introduction

Many coverage criteria for software testing, such as statement and path coverage, treat each statement as a single node. To take into account the decisions within the nodes, decision coverage (DC) and condition coverage (CC) have earlier been proposed [14], but they are well-known to be inadequate for testing software embedded with complex logical decisions, such as those commonly found in safety-critical software [5, 6, 8]. In such cases, it is imperative that the logical decisions be adequately tested for the occurrence of plausible faults.

In recent years, more sophisticated coverage criteria have been advocated, including the condition/decision coverage (C/DC), modified condition/decision coverage (MC/DC) [6, 8, 10] and the MUMCUT criteria [4, 18]. In particular, compliance of MC/DC has been mandated for more than a decade in the aviation industry for the approval of airborne software. On the other hand, the MUMCUT criterion [4] has been proved to guarantee the detection of certain faults in logical decisions in irredundant disjunctive normal form [2, 3]. However, its effectiveness in detecting faults in other logical decisions has not been clearly understood.

Many studies of coverage criteria related to logical decisions have focused on the generation of test cases [2, 4, 6, 15, 17] or the reduction of the size of test sets [10, 18]. There were a few empirical reports on the fault-detecting ability of test sets satisfying these criteria, but they were mainly concerned with the detection of five classes of fault, namely, expression negation fault (ENF), variable negation fault (VNF), variable reference fault (VRF), operator reference fault (ORF) and associative shift fault (ASF) [5, 11, 17]. Many other classes of faults have not been considered, such as the insertion of superfluous (extra) conditions [2, 9, 14, 15], which have been hypothesized specifically for modeling plausible faults in logical decisions.

Towards the goal of filling this gap, this paper compares several coverage criteria by both theoretical analysis and empirical evaluation. Section 2 presents the terminologies and definitions. Section 3 formally analyses the subsume relations of these coverage criteria. Sections 4 and 5 compare theoretically and empirically the fault-detecting ability of test sets satisfying these coverage criteria. Section 6 concludes this paper.

2 Preliminaries

2.1 Notation and Terminology

A **Boolean expression** is one which evaluates to either **FALSE** (0) or **TRUE** (1). A **condition** is a Boolean expression with no Boolean operators. A **logical decision**, or simply a **decision**, is a Boolean expression composed of conditions and zero or more Boolean operators. **We**
denote the Boolean operators AND, OR, and NOT by “·”, “+” and “¬”, respectively. Whenever it is sufficiently clear, the “·” symbol will usually be omitted. Thus, the expression \( a \) OR \( b \) AND NOT \( c \) can be succinctly represented as \( a + b \overline{c} \).

A condition within a decision is denoted by a letter (variable) such as \( a, b, c, \ldots \), which may represent either a simple Boolean variable such as “Running” (indicating whether or not the engine of an automobile is running) in a Cruise Control System [1], or a relational expression such as “rom->paddles-deployed == 1” [8].

A variable \( a \) in a decision may occur as a positive literal \((a)\) or a negative literal \((\overline{a})\), such as in the first and second term of \( S = ab + \overline{a}b \), respectively.

A test case for a decision \( S \) in \( n \) variables is a vector \( \vec{t} = (t_1, \ldots, t_n) \), where \( t_i \) is the value assigned to the \( i \)-th variable and \( t_i \in \mathbb{B} = \{0, 1\} \). We denote by \( S(\vec{t}) \) the outcome of \( S \) when its variables are respectively assigned the values \( t_1, \ldots, t_n \). Two decisions \( S_1 \) and \( S_2 \) in \( n \) variables are said to be equivalent, denoted by \( S_1 = S_2 \), if \( S_1(\vec{t}) = S_2(\vec{t}) \) for all test cases \( \vec{t} \in \mathbb{B}^n \). A test case \( \vec{t} \) is said to distinguish two decisions \( S_1 \) and \( S_2 \) if \( S_1(\vec{t}) \neq S_2(\vec{t}) \).

A decision \( S \) in \( n \) variables can always be written in disjunctive normal form (DNF) as a sum of products, or equivalently in its irredundant disjunctive normal form (IDNF), which is a DNF such that none of its terms nor literals may be omitted without changing the outcomes of the decision for some combination of values of its conditions. Although a decision may be written in IDNF, DNF or non-DNF, the first two forms are much more common in practice [5].

A test case \( \vec{t} \) for a decision \( S = p_1 + \ldots + p_m \) in DNF is said to be a true point (respectively a false point) if \( S(\vec{t}) = 1 \) (respectively \( S(\vec{t}) = 0 \)). If \( p_i(\vec{t}) = 1 \) but \( p_j(\vec{t}) = 0 \) for every \( j \neq i \), then \( \vec{t} \) is said to be a unique true point (UTP) of the \( i \)-th term \( p_i \) of \( S \).

Let the \( i \)-th term of a decision \( S \) in DNF be \( p_i = x_{i1} \cdot x_{i2} \cdot \ldots \cdot x_{ik} \), where \( x_{ij} \) is the \( j \)-th literal in \( p_i \) and \( k_i \) be the number of literals in \( p_i \). We use \( p_i, j = x_{i1} \cdot x_{i2} \cdot \ldots \cdot x_{ik} \) to denote the term obtained from \( p_i \) by negating its \( j \)-th literal \( x_{ij} \). A test case \( \vec{f} \) is said to be a near false point (NFP) of the literal \( x_{ij} \) of \( p_i \) if \( S(\vec{f}) = 0 \) but \( p_{i,j}(\vec{f}) = 1 \).

### 2.2 Faults in Logical Expressions

Observations of real faults in [7,8] are consistent with the common intuition that typical programmer errors include missing or extra conditions and paths, and the use of incorrect operators or operands [14,15]. Insofar as logical decisions are concerned, these errors translate to the omission, insertion or incorrect reference of conditions and Boolean operators. As such, we shall consider the following classes of simple faults for logical decisions that are written in DNF. For the purpose of illustration, let the correct expression be \( S = ab + \overline{c}d + e \).

- **Expression Negation Fault**\(^1\) (ENF):- The entire expression, or a Boolean sub-expression thereof, is replaced by its negation. For example, \( S \) becomes \( \overline{ab + cd + e} \).
- **Term Negation Fault** (TNF):- A term is replaced by its negation. For example, \( S \) becomes \( \overline{ab + cd + e} \).
- **Term Omission Fault** (TOF):- A term is omitted. For example, \( S \) becomes \( \overline{cd + e} \).
- **Operator Reference Fault** (ORF):- It refers to either a disjunctive ORF (ORF[+]) which replaces an AND operator (+) by AND (·), or a conjunctive ORF (ORF[-]) which replaces an AND operator (-) by OR (+). For example, \( S \) becomes \( ab \cdot \overline{cd + e} \) or \( a + b + \overline{cd + e} \), respectively.
- **Literal Negation Fault** (LNF):- A literal is replaced by its negation. For example, \( S \) becomes \( ab + cd + e \) when the first literal \( a \) is negated. This fault is often referred to as a Variable Negation Fault\(^2\) (VNF) in many related studies, such as in [11,17].
- **Literal Omission Fault** (LOF):- A literal is omitted. For example, \( S \) becomes \( b + \overline{cd + e} \) when the first literal \( a \) is omitted.
- **Literal Insertion Fault** (LIF):- A literal is inserted into a term of the expression. For example, \( S \) becomes \( abc + \overline{cd + e} \) when the literal \( c \) is inserted into the first term.
- **Literal Reference Fault** (LRF):- A literal is replaced by another literal. For example, \( S \) becomes \( cb + \overline{cd + e} \) when the first literal \( a \) is replaced by \( c \). This fault is often referred to as a Variable Reference Fault\(^2\) (VRF) in many related studies, such as in [11,17].
- **Stuck-At Fault** (STF):- It refers to either a stuck-at-0 fault (STF[0]) or a stuck-at-1 fault (STF[1]), which causes the value of a literal to be stuck at 0 or 1, respectively. For example, when the literal \( a \) is stuck at the value 0, \( S \) will become \( 0 \cdot \overline{b + \overline{cd + e}} \), which simplifies to \( \overline{cd + e} \); and when \( a \) is stuck at 1, \( S \) will become \( 1 \cdot \overline{b + \overline{cd + e}} \), which simplifies to \( b + \overline{cd + e} \).

\(^1\)This paper adopts a slightly generalised and more useful definition of expression negation fault (ENF) than as appeared in [3,4], where ENF was defined as causing the negation of the entire expression.

\(^2\)Note that a variable negation (or reference) fault VNF (or VRF) is also used in some studies (say, in [12]) to mean the fault which replaces every occurrence of the variable in the expression by its negation (or by another variable). This can be different from the literal faults referred to in this paper when the variable occurs more than once in the expression. However, so far we are not aware of any empirical studies that deal with the detection of faults which affect every occurrence of a variable.
Except for TNF and TOF, all of the above fault classes apply equally well to logical decisions that are not written in DNF. Moreover, for non-DNF expressions, faults may also occur in the brackets or in the precedence of evaluation of the conditions, and so the following class of fault will also be considered.

- **Associative Shift Fault (ASF):** The associativity of terms is changed once. For example, the non-DNF expression \( a(b + c) \) is replaced by \((ab) + c\).

In general, a faulty decision resulting from one of these faults may happen to be equivalent to another due to a different fault. For example, if \( S = ab + cd + e \) is the correct decision, then the faulty decision \( S' = cd + e \) may be the result due to TOF when the term \( ab \) is omitted, or due to STF[0] when the literal \( a \) is stuck at the value 0.

### 2.3 Coverage Criteria for Testing Decisions

The definitions of decision coverage (DC), condition coverage (CC) and condition/decision coverage\(^3\) (C/DC) all include the requirement that “every point of entry and exit in the program has been invoked at least once” [6, 14]. The other parts of their definitions are as follows:

- **DC (Decision Coverage):** every decision has taken on all possible outcomes at least once;
- **CC (Condition Coverage):** every condition has taken on all possible outcomes at least once;
- **C/DC (Condition/Decision Coverage):** every condition in a decision has taken on all possible outcomes at least once, and every decision in the program has taken on all possible outcomes at least once;
- **MC/DC (Modified Condition/Decision Coverage):** every condition in a decision has taken on all possible outcomes at least once, and every decision has taken on all possible outcomes at least once, and each condition has been shown to independently affect the decision’s outcome. A condition is shown to independently affect a decision’s outcome by varying just that condition while holding fixed all other possible conditions.

Given a logical decision in IDNF, the MUMCUT criterion requires that the test set \( T \) satisfies *all* of the three criteria: **MUTP, MNFP** and **CUTPNFP**, defined as follows [2–4].

- **MUTP (Multiple Unique True Point):** for every \( i \), \( T \) contains unique true points of the \( i \)-th term \( p_i \) such that all possible truth values (that is, 0 and 1) of every variable not occurring in \( p_i \) are covered.
- **MNFP (Multiple Near False Point):** for every \( i \) and \( j \), \( T \) contains near false points of the \( j \)-th literal of the \( i \)-th term \( p_i \) such that all possible truth values (that is, 0 and 1) of every variable not occurring in \( p_i \) are covered.
- **CUTPNFP (Multiple Unique True Point):** for every \( i \) and \( j \), as far as possible, \( T \) contains a unique true point \( \bar{t} \) of the \( i \)-th term \( p_i \) and a near false point \( \bar{f} \) of the \( j \)-th literal of \( p_i \) such that \( \bar{t} \) and \( \bar{f} \) differ only in the corresponding truth value of the \( j \)-th literal of \( p_i \).

When the decision \( S \) is not in IDNF, it can always be transformed into an equivalent expression in IDNF before test sets are generated to satisfy the MUMCUT criterion.

### 3 Subsumption Analysis

#### 3.1 The Subsume Relation

**Definition 1**  Let \( C \) be a coverage criterion. A test set \( T \) is said to be **\( C \) adequate** if it satisfies \( C \). In this case, \( T \) is also called a **\( C \) adequate test set**, or simply a **\( C \) test set**.

For instance, a MC/DC test set is one which satisfies the MC/DC criterion and hence is MC/DC adequate.

One common way of theoretically comparing coverage criteria is to consider the subsume relation [6].

**Definition 2**  A coverage criterion \( C_1 \) is said to subsume criterion \( C_2 \) if every \( C_1 \) adequate test set is also \( C_2 \) adequate.

The subsume relation basically compares the relative difficulty of satisfaction of coverage criteria: if criterion \( C_1 \) subsumes criterion \( C_2 \), then \( C_1 \) is harder to satisfy than \( C_2 \). We shall analyse the fault-detecting ability of MC/DC and MUMCUT test sets later in Section 4.

#### 3.2 A Subsumption Hierarchy

The subsume relations among DC, CC, C/DC and MC/DC follow directly from their definitions as shown in [6].

**Theorem 1**  The following relations hold [6]:

(a) C/DC subsumes both DC and CC  
(b) MC/DC subsumes C/DC

The subsume relations among MUMCUT, MUTP, MNFP and CUTPNFP also follow directly from their definitions given in Section 2.3 above (see also [4]).

**Theorem 2**  MUMCUT subsumes MUTP, MNFP and CUTPNFP.

The following lemma states an important property which will be needed for proving subsequent theorems.

\(^3\)Myers [14] used the term “decision/condition coverage” instead.
Lemma 1 (Theorem 1 in [3]) Let $S = p_1 + \cdots + p_m$ be a logical decision in IDNF. Then for every $i$ and $j$, there exists at least one UTP of the $i$-th term $p_i$ of $S$, and at least one NFP of the $j$-th literal of $p_j$ of $S$.

The next lemma follows directly from Lemma 1 and the definition of MUMCUT.

Lemma 2 Let $S = p_1 + \cdots + p_m$ be a logical decision in IDNF. Then every MUMCUT test set $T$ contains at least one UTP of the $i$-th term $p_i$ of $S$, and at least one NFP of the $j$-th literal of $p_j$ of $S$.

Theorem 3 MUMCUT subsumes DC.

Proof Let $S$ be any logical decision, and $S_0$ be an IDNF expression such that $S_0 \equiv S$. Let $T$ be any MUMCUT test set generated from $S_0$. By Lemma 2, $T$ contains at least one UTP $\vec{t}$ of $p_q$ of $S_0$, and at least one NFP $\vec{t}'$ of the literal $x$ of $p_q$ of $S_0$. Since $p_q$ contains the literal $x$ and $p_q(\vec{t}) = 1$, the value of $x$ in $\vec{t}$ must be 1. Also, since $\vec{t}'$ is a NFP of $x$ in $p_q$ of $S_0$, $p_q \bar{x}(\vec{t}') = 1$. Hence, the value of $x$ in $\vec{t}'$ must be 0.

Case (ii) $S_0$ has a term $p_q$ which contains the literal $\bar{x}$. By Lemma 2, $T$ contains at least one UTP $\vec{t}$ of $p_q$ of $S_0$, and at least one NFP $\vec{t}'$ of the literal $\bar{x}$ of $p_q$ of $S_0$. Since $p_q$ contains the literal $\bar{x}$ and $p_q(\vec{t}) = 1$, the value of $x$ in $\vec{t}$ must be 0. Also, since $\vec{t}'$ is a NFP of $\bar{x}$ of $p_q$ of $S_0$, $p_q \bar{x}(\vec{t}') = 1$. Hence, the value of $\bar{x}$ in $\vec{t}'$ must be 1.

Case (iii) Neither $x$ nor $\bar{x}$ occurs in $S_0$. Let $p_q$ be any term of $S_0$. Since variable $x$ does not occur in $p_q$, the MUTP criterion requires that at least two UTPs, which we denote by $\vec{t}_0$ and $\vec{t}_1$, of $p_q$ of $S_0$ be selected such that the value of $x$ is 0 and 1, respectively. This is always possible, because in this case, the value of $x$ will not affect the value of any term in $S_0$.

In all cases, $T$ contains the test cases $\vec{t}_0$ and $\vec{t}_1$ in which the value of $x$ is 0 and 1, respectively. This is true for every variable $x$ in $S$. Therefore, $T$ satisfies CC.

Theorem 5 MUMCUT subsumes C/DC.

Proof It follows directly from Theorems 3 and 4.

4 Fault Detection Analysis

Some coverage criteria $C$ have the desirable property that every $C$ test set will always detect certain faults.

Definition 3 Let $S$ be a logical decision and $F$ be a fault. A test set $T$ is said to detect $F$ if for every faulty decision $S'$ resulting from $S$ due to a single instance of $F$ such that $S'$ is not equivalent to $S$, there exists a test case $\vec{t} \in T$ such that $\vec{t}$ distinguishes $S$ and $S'$ (that is, $S(\vec{t}) \neq S'(\vec{t})$).

Definition 4 A coverage criterion $C$ is said to guarantee to detect a fault $F$ if every $C$ adequate test set detects $F$.

In general, although a logical decision may appear in IDNF, DNF or other forms, the first two forms are undoubtedly the most common in practice. For instance, Chilenski [5] has extracted more than 20,000 logical decisions from the airborne software of five different Line Replaceable Units (LRUs, also known as black boxes) in five different systems across two airplane models. Over 95% of the occurrences of the decisions tabulated in [5] are in DNF, and over 99.5% of these DNF decisions are also in IDNF. Clearly, IDNF decisions form a very important subclass of all decisions used in practice.

In the rest of this section, we shall analyze theoretically which of the faults that MC/DC and MUMCUT guarantee to detect in logical decisions written in IDNF.

4.1 The MC/DC Criterion

The MC/DC criterion has been well-known for more than two decades [6], and it is only until fairly recently that a few experiments on its fault-detecting ability have been reported [5, 8, 10].
In [12], Kuhn proved a fault class hierarchy which shows that ENF is easier to detect than variable negation fault and variable reference fault. Following up Kuhn’s work, we have recently formally established an extended fault class hierarchy among the faults in IDNF expressions [13], showing that ENF and TNF are the two faults that are easiest to detect.

We find that MC/DC only guarantees to detect ENF and TNF, but none of the other faults in IDNF decisions.

Theorem 6 Let S be a logical decision in IDNF and S′ be an expression resulting from S due to an ENF. If ̇f is a false point of S, then ̇S( ̇f) ̸= S′( ̇f).

Proof First, if the entire expression S is negated, then S′ ̸≡ S, and obviously S′( ̇f) ̸= S( ̇f). Otherwise, let S ̸≡ S1 + S2 where S1 is the sub-expression of S to be negated due to an ENF, and S2 is the remaining sub-expression of S. Then S′ ̸≡ S1 + S2. Since S( ̇f) = 0, S1( ̇f) = S2( ̇f) = 0. Hence S′( ̇f) = S1( ̇f) + S2( ̇f) = 1 ̸= S( ̇f).

Corollary 1 Let S be a logical decision in IDNF and S′ be an expression resulting from S due to a TNF. If ̇f is a false point of S, then ̇S( ̇f) ̸= S′( ̇f).

Proof The proof follows from that of Theorem 6 when the expression to be negated is a single term of S.

Theorem 7 If a logical decision S is in IDNF, then MC/DC guarantees to detect both ENF and TNF.

Proof By definition, every MC/DC test set T contains at least one false point of S. By Theorem 6 and Corollary 1, T will detect both ENF and TNF.

Theorem 8 MC/DC does not guarantee to detect TOF, ORF, LNF, LOF, LIF, LRF and STF even when the logical decision S is in IDNF.

Proof Consider the logical decision S = adc + be + ace + adg + aeg + bf and the faulty decisions that may result from S due to a single instance of each of TOF, ORF, LNF, LOF, LIF, LRF and STF as shown in Table 1. Let T be the test set { ̇t1, ̇t2, ̇t3, ̇t4} as shown in Table 2, where S( ̇ti) = 1 for i = 1, . . . , 4 and S( ̇fj) = 0 for j = 1, . . . , 5. It is clear from Table 2 that, for each condition, there is a true point ̇ti and a corresponding false point ̇fj in T such that the pair ( ̇ti, ̇fj) shows that the condition independently affects the outcome of S. Hence T is MC/DC adequate. However, both S and each of the faulty decisions in Table 1 evaluate to the same outcome for every test case in T. Therefore, T cannot distinguish S from any of the faulty decisions in Table 1.

Table 1: Decisions used in the proof of Theorem 8

<table>
<thead>
<tr>
<th>Correct decision</th>
<th>Faulty decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>S = adc + be + ace + adg + aeg + bf</td>
<td>S_{TOF} = be + ace + adg + aeg + bf</td>
</tr>
<tr>
<td></td>
<td>S_{ORF} = adc \cdot be + ace + adg + aeg + bf</td>
</tr>
<tr>
<td></td>
<td>S_{LNF} = \neg (adc + be + ace + adg + aeg + bf)</td>
</tr>
<tr>
<td></td>
<td>S_{LOF} = dc + be + ace + adg + aeg + bf</td>
</tr>
<tr>
<td></td>
<td>S_{LIF} = bc + be + ace + adg + aeg + bf</td>
</tr>
<tr>
<td></td>
<td>S_{LRF} = b\cdot dc + be + ace + adg + aeg + bf</td>
</tr>
<tr>
<td></td>
<td>S_{STF} = 0 \cdot dc + be + ace + adg + aeg + bf</td>
</tr>
<tr>
<td></td>
<td>S_{STF[1]} = 1 \cdot dc + be + ace + adg + aeg + bf</td>
</tr>
</tbody>
</table>

Table 2: Test cases used in the proof of Theorem 8

<table>
<thead>
<tr>
<th>Condition shown by the pair ( ̇ti, ̇fj) to individually affect the outcome of S = abc + be + ace + aeg + bf</th>
<th>True points</th>
<th>False points</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1: 1001000 a 0001000 f1</td>
<td>1001000</td>
<td>f1</td>
</tr>
<tr>
<td>t2: 1000100 e 1000000 f2</td>
<td>1000100</td>
<td>f2</td>
</tr>
<tr>
<td>t3: 1010101 c 1001011 f3</td>
<td>1001011</td>
<td>f3</td>
</tr>
<tr>
<td>t4: 0100010 b 0000010 f4</td>
<td>0100010</td>
<td>f4</td>
</tr>
<tr>
<td>t5: 0100010 f 0100000 f5</td>
<td>0100000</td>
<td>f5</td>
</tr>
</tbody>
</table>

4.2 The MUMCUT Criterion

The MUMCUT criterion, which is an integration of the MUTP, MNFP and CUTPNFP criteria, was recently specifically designed to detect faults in IDNF Boolean expressions [2–4]. Chen and Lau [3] have proved that the MUMCUT criterion guarantees the detection of the following faults in IDNF decisions4: TOF, ORF, LNF, LOF, LIF and LRF.

To complete the analysis, we now prove that MUMCUT also guarantees to detect ENF, TNF and STF when the logical decision is in IDNF.

Theorem 9 If a logical decision S is in IDNF, then MUMCUT guarantees to detect both ENF and TNF.

Proof By definition, every MUMCUT test set T contains at least one near false point of S. By Theorem 6 and Corollary 1, T will detect both ENF and TNF.

---

4In [3], Chen and Lau have also found that MUMCUT will detect a special subclass of ENF which causes the entire Boolean expression to be implemented as its negation. However, in line with other related work such as [12], this paper considers the more general form of ENF in which a sub-expression is replaced by its negation.
Theorem 10 Let $S$ be a logical decision in IDNF and $S'$ be the resulting expression due to a STF[0] when the literal $x^q_j$ of the term $p_q$ of $S$ is stuck at the value 0. If $\bar{\bar{r}}$ is a UTP of the term $p_q$ of $S$, then $S(\bar{\bar{r}}) \neq S'(\bar{\bar{r}})$.

**Proof** Since $\bar{\bar{r}}$ is a UTP of the term $p_q$ of $S$, $S(\bar{\bar{r}}) = 1$. When the literal $x^q_j$ of the term $p_q$ is stuck at the value 0, $p_q \equiv 0$. If $p_q$ is the only term in $S$, then $S' \equiv 0$ and so $S(\bar{\bar{r}}) \neq S'(\bar{\bar{r}})$.

Otherwise, if there is more than one term in $S$, then $S'$ is equivalent to the expression resulting from $S$ with the term $p_q$ omitted. Since $\bar{\bar{r}}$ is a UTP of $p_q$ of $S$, $p_q(\bar{\bar{r}}) = 1$ and for all $i \neq q$, $p_i(\bar{\bar{r}}) = 0$. As $S$ contains the term $p_q$ and $S'$ does not, $S(\bar{\bar{r}}) = 1$ and $S'(\bar{\bar{r}}) = 0$. $\square$

Theorem 11 Let $S$ be a logical decision in IDNF and $S'$ be the resulting expression due to a STF[1] when the literal $x^q_j$ of the term $p_q$ of $S$ is stuck at the value 1. If $\hat{\bar{\bar{r}}}$ is a NFP of the literal $x^q_j$ of the term $p_q$ of $S$, then $S(\hat{\bar{\bar{r}}}) \neq S'(\hat{\bar{\bar{r}}})$.

**Proof** Since $\hat{\bar{\bar{r}}}$ is a NFP of the literal $x^q_j$ of the term $p_q$ of $S$, by definition of NFP, $S(\hat{\bar{\bar{r}}}) = 0$ and $p_{q,j}(\hat{\bar{\bar{r}}}) = 1$, where $p_{q,j} = x^q_1 \cdot x^q_2 \cdot \ldots \cdot x^q_k$. Thus, $x^q_j(\hat{\bar{\bar{r}}}) = 0$ and for all $i \neq j$, $x^q_i(\hat{\bar{\bar{r}}}) = 1$. When the literal $x^q_j$ is stuck at the value 1, $p_q$ becomes $p'_q$ where $p'_q = x^q_1 \cdot \ldots \cdot x^q_{j-1} \cdot 1 \cdot x^q_{j+1} \cdot \ldots \cdot x^q_k$. Hence $p'_q(\hat{\bar{\bar{r}}}) = 1$, and $S'(\hat{\bar{\bar{r}}}) = 1 \neq S(\hat{\bar{\bar{r}}})$. $\square$

Theorem 12 If a logical decision $S$ is in IDNF, then MUMCUT guarantees to detect STF.

**Proof** Theorem 12 follows immediately from Theorems 10 and 11 because, by Lemma 2, a MUMCUT test set contains at least one UTP of every term in $S$ and at least one NFP of every literal of every term in $S$.

To summarize, MUMCUT guarantees to detect all faults described in Section 2.2 that are applicable to logical decisions in IDNF.

Theorem 13 If a logical decision $S$ is in IDNF, then MUMCUT guarantees to detect ENF, TNF, TOF, ORF, LNF, LOF, LIF, LRF and STF.

5 Empirical Analysis

5.1 Research Questions

Our analysis in the previous sections have shown that

- both MUMCUT and MC/DC subsume C/DC, DC and CC, and
- MUMCUT guarantees to detect all the classes of fault defined in Section 2.2 for IDNF decisions, but
- MC/DC only guarantees to detect ENF and TNF.

Unfortunately, theoretical analysis presently does not provide enough insight on how common these criteria enable other faults to be detected.

Given the practical importance of MC/DC and other coverage criteria, it is desirable to better understand not only their fault-detecting ability in the worst case, but also in the common cases. Towards this goal, we performed an empirical study with these research questions in mind:

1. What are the costs, in terms of the number of test cases required, of satisfying the DC, CC, C/DC, MC/DC and MUMCUT criteria for logical decisions that occur in practice? This helps to assess the cost/effectiveness tradeoff of using these criteria.
2. How effective are the DC, CC, C/DC and MC/DC criteria in detecting faults in DNF decisions that occur in practice?
3. How effective are the DC, CC, C/DC, MC/DC and MUMCUT criteria in detecting faults in non-DNF decisions that occur in practice?

5.2 Subject Logical Decisions

The subjects of our experiments are samples of logical decisions extracted from the software of Line Replaceable Units (also known as black boxes), which belonged to five different airborne systems across two different airplane models [5]. From this collection, we sampled 20 DNF and 20 non-DNF decisions that included, as far as feasible, logical decisions with different numbers of variables.

Thus, we experimented with a total of 40 logical decisions, half of which are in DNF and half are not. The DNF decisions were identified as D01, D02, ..., D20, and the non-DNF decisions as N01, N02, ..., N20, respectively. We have verified that all the sample DNF decisions are also in IDNF. The number of variables in each of these 40 decisions ranges between 3 and 16.

For each criterion $C$, we generated as many $C$ adequate test sets as possible for each decision $S$, unless the number exceeds 1000. In the latter case, we generated a random sample of 1000 test sets that are $C$ adequate for $S$. Then we generated all expressions that can be formed by introducing to each of the 40 decisions a single instance of a fault of the classes defined in Section 2.2, and used only those resulting expressions that are not equivalent to the original decisions. Finally, we recorded which test set can distinguish an original decision from its faulty versions due to each class of fault and computed the statistics.

5.3 Results and Observations

We observe that DC can be satisfied by using a true point and a false point, while CC can be satisfied by two test cases $\bar{\bar{r}}_1$ and $\bar{\bar{r}}_2$ such that $\bar{\bar{r}}_1$ is the complement of $\bar{\bar{r}}_2$. Hence DC or CC always requires only two test cases.
If a CC adequate test set $T$ of size two happens to satisfy DC as well, then $T$ is already C/DC adequate. Otherwise, either both test cases in $T$ are true points or both are false points, and adding a false point or a true point, respectively, to $T$ will render it C/DC adequate. Hence C/DC requires only either two or three test cases.

Table 3 shows the size of test sets generated to satisfy each criterion, averaged over the DNF and non-DNF decisions, respectively. The two sets of averages are very similar; except that MUMCUT requires notably larger test sets for non-DNF decisions than for DNF decisions. On average, a MUMCUT test set is about 2 or 3 times as large as a MC/DC test set, and the latter is much larger than a DC, CC or C/DC test set.

Tables 4 and 5 show the average percentage of detected faults for DNF and non-DNF decisions, respectively.

Since the sample DNF decisions are also in IDNF, MUMCUT detects all the faults in Table 4, and MC/DC detects all ENFs and TNFs, as proved in Section 4. For our batch of DNF logical decisions, MC/DC test sets catch all of the ORFs (though it is not guaranteed in theory), and between 67% and 98% of each other class of faults. Averaged over all classes of fault under study, the percentage of faults detected is about 93%. Thus, MC/DC is clearly very cost-effective; it possesses very good fault-detecting ability for all faults except LIF, yet generally it does not require a very large test set.

As found in previous fault class analyses, both ENF and TNF are among the easiest faults to detect [12, 13, 16]. Table 4 shows that, for DNF decisions, using only two or three test cases, both DC and C/DC catch all ENFs and TNFs, and CC detects 81% and 60% of them, respectively. For other faults, DC, CC and C/DC are clearly much less effective than MC/DC or MUMCUT. In particular, MC/DC is about 6 to 7 times more effective than CC in detecting LOF, LIF, LRF and STF in DNF decisions.

Interestingly, although both DC and CC test sets require two test cases, DC seems to detect a significantly higher percentage of faults on average than CC for all classes of faults in DNF decisions except LIF and TOF.

Comparing Tables 4 and 5, for each criterion, the overall average percentage of faults detected does not differ very much between DNF and non-DNF decisions, despite some variations between individual faults. Thus, the five criteria can be arranged in decreasing order of their overall fault-detecting ability as follows: MUMCUT > MC/DC > C/DC > DC > CC. This order also holds when comparing the averages for individual faults, except that CC is slightly better than DC for LIF and TOF in DNF decisions and ASF in non-DNF decisions.

For non-DNF decisions, ENF is still the easiest fault to detect, but unlike DNF decisions, none of DC, CC or C/DC test sets can catch all ENFs. Although MUMCUT was designed to detect faults in DNF, Table 5 shows that it is also extremely effective in detecting faults in non-DNF decisions. It detects all faults of the classes ENF, ORF, LNF, LOF and STF in this experiment. Only very few faulty decisions due to LIF, LRF or ASF escape the detection of the MUMCUT test sets. Thus, if no fault is found by a MUMCUT test set, it would provide a much higher confidence of the correctness of the logical decisions than test sets satisfying the other four criteria.

In this experiment, for non-DNF decisions, MC/DC test sets catch 77% of LIFs, 85% of ASFs and more than 90% of all other faults. In contrast, DC, CC and C/DC test sets are far less effective. While MC/DC (respectively MUMCUT) test sets can reveal over 90% (respectively 99%) of the faults in all logical decisions under study, more than half of these faults will escape the detection by DC, CC and C/DC test sets.

<table>
<thead>
<tr>
<th>Fault class</th>
<th>Average % of faults detected (D01 – D20)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ENF</td>
<td>100.0</td>
<td>80.5</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>TNF</td>
<td>100.0</td>
<td>60.2</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>TOF</td>
<td>14.6</td>
<td>16.1</td>
<td>24.2</td>
<td>96.9</td>
<td>100.0</td>
</tr>
<tr>
<td>ORF</td>
<td>41.1</td>
<td>33.0</td>
<td>51.7</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>LNF</td>
<td>45.6</td>
<td>25.3</td>
<td>53.6</td>
<td>98.4</td>
<td>100.0</td>
</tr>
<tr>
<td>LOF</td>
<td>21.6</td>
<td>13.5</td>
<td>26.1</td>
<td>90.8</td>
<td>100.0</td>
</tr>
<tr>
<td>LIF</td>
<td>10.1</td>
<td>11.0</td>
<td>16.5</td>
<td>67.0</td>
<td>100.0</td>
</tr>
<tr>
<td>LRF</td>
<td>25.9</td>
<td>13.9</td>
<td>31.4</td>
<td>93.5</td>
<td>100.0</td>
</tr>
<tr>
<td>STF</td>
<td>25.1</td>
<td>13.3</td>
<td>31.1</td>
<td>93.8</td>
<td>100.0</td>
</tr>
<tr>
<td>Ave.</td>
<td>42.7</td>
<td>29.6</td>
<td>48.3</td>
<td>93.4</td>
<td>100.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fault class</th>
<th>Average % of faults detected (N01 – N20)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ENF</td>
<td>67.4</td>
<td>59.5</td>
<td>72.8</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>ORF</td>
<td>43.2</td>
<td>35.3</td>
<td>51.5</td>
<td>99.6</td>
<td>100.0</td>
</tr>
<tr>
<td>LNF</td>
<td>38.4</td>
<td>30.9</td>
<td>48.3</td>
<td>99.7</td>
<td>100.0</td>
</tr>
<tr>
<td>LOF</td>
<td>18.1</td>
<td>17.3</td>
<td>25.0</td>
<td>92.4</td>
<td>100.0</td>
</tr>
<tr>
<td>LIF</td>
<td>21.5</td>
<td>20.2</td>
<td>30.7</td>
<td>76.9</td>
<td>99.9</td>
</tr>
<tr>
<td>LRF</td>
<td>21.8</td>
<td>15.5</td>
<td>28.7</td>
<td>94.7</td>
<td>100.0</td>
</tr>
<tr>
<td>STF</td>
<td>23.3</td>
<td>17.1</td>
<td>31.6</td>
<td>95.4</td>
<td>100.0</td>
</tr>
<tr>
<td>ASF</td>
<td>40.6</td>
<td>42.3</td>
<td>55.2</td>
<td>85.1</td>
<td>99.6</td>
</tr>
<tr>
<td>Ave.</td>
<td>34.3</td>
<td>29.8</td>
<td>43.0</td>
<td>93.0</td>
<td>99.9</td>
</tr>
</tbody>
</table>

5 To be more precise, the percentage is 99.98 and not exactly 100.
6 Summary and Conclusion

Logical decisions are widely used in software. Although there are many well-known coverage criteria for testing logical decisions, the fault-detecting ability of test sets satisfying these criteria has been largely unclear. To contribute to this body of knowledge, the present paper performs both formal and empirical analysis to compare the DC, CC, C/DC, MC/DC and MUMCUT criteria.

Formally, we have proved that MUMCUT subsumes C/DC, CC and DC. For the important class of IDNF logical decisions, which are the majority of decisions in practical use, we proved that MC/DC guarantees to detect ENF and TNF, while MUMCUT guarantees to detect all of ENF, TNF, TOF, ORF, LNF, LOF, LIF, LRF and STF.

Empirically, we experimented with 20 DNF and 20 non-DNF logical decisions that were used in real software systems. We find that for all of them, MUMCUT detects more faults than MC/DC, which in turn detects many more faults than C/DC, CC and DC. Further work is currently underway to investigate to what extent the larger number of test cases required by the more stringent criteria contributes to their effectiveness of detecting faults. We are also working further on the formal analysis with an aim to uncovering more theoretical properties and relations of these and other criteria.

Finally, in a nutshell, it is evident that MC/DC is cost-effective, as it represents a good balance of testing cost and fault-detecting ability. However, it may still miss some faults that will always (if the logical decision is in IDNF) or at least almost always be caught by a MUMCUT test set. When ultra-critical software is concerned, the guarantee of MUMCUT in the detection of all single instances of the faults defined in Section 2.2 for IDNF logical decisions will be particularly helpful.

Acknowledgement

The authors thank Isaac Yeung and Alfred Chan for working on the empirical study described in this paper.

References


