CHAPTER 5

EXCEPTIONS IN HIERARCHY AND ALTERNATE STRATEGY

5.1 INTRODUCTION

In the last chapter, test expressions to generate test sets for various faults from Literal Reference Fault were proposed and the hierarchy of fault classes was analyzed. Continuing with the analysis of the hierarchy of fault classes, it was observed that in some cases, the faulty specification was equivalent to the original specification. That is, the faulty specification is another form of the original specification. Hence, the fault condition for the particular fault will not generate any test cases. One of the important tasks of the software tester is to ascertain that the implementation is done exactly as per the specification. Under these circumstances, the hierarchy of fault classes does not hold. In this chapter, the Null Fault is defined and a method to detect the Null Fault is proposed. Some exceptions in the hierarchy of fault classes are explained. Further, an alternate strategy is proposed that would help to derive test cases for various faults from Literal Negation Fault. The need for going to the higher fault model is also explained.

Section 5.2 defines the Null Fault. Section 5.3 discusses the exceptions that surfaced during the analysis of fault classes. Section 5.4 gives the test expressions to generate test sets for various faults using Literal Negation Fault. Section 5.5 discusses the reasons to go for Literal Negation Fault. The results are applied to real life Traffic Alert and Collision Avoidance Subsystem (TCAS II) specifications in Section 5.6. Section 5.7 gives a comparison of other test set generation methods with the proposed method. Section 5.8 discusses the significance of the results obtained.
5.2 NULL FAULT

Definition: If the original specification $S$ is implemented in its equivalent form, then it is termed as Null Fault. In this case, the faulty specification and the original specification evaluate to the same truth value for all the input combinations.

Specification provides valuable information for testing. Considering the information from formal specification enables testing intended behaviour as well as actual functionality. The implementer may implement the specification incorrectly or neglect an aspect of a problem. There are situations in which the incorrect implementation of the specification can be another form of the specification. It is necessary to ascertain that the specification is implemented correctly. Even though the faulty implementation of the specification is equivalent to the actual specification, it is essential to detect such faults because if a part of the specification is used in another part of the system, it may lead to serious errors, particularly in Safety Critical Systems. Many researchers have discussed the above situation.

Woodward (1993) mentioned that the basic idea of (strong) mutation testing is to make many small changes, one at a time, to a given program. Then an attempt is made to provide test data, which can distinguish each of these so-called mutants from the original program. If a mutant gives a different outcome to the original with the test data, it is said to be dead. As it has been distinguished from the original, it need no longer be considered. If, on the other hand, a mutant gives the same outcome as the original with the test data, then it is said to be live. Further investigations should be able to reveal whether the test data could be enhanced to kill this live mutant or whether test data could never be constructed to kill the mutant. In the former case, the original test data can be considered to have been inadequate, and hence have been improved by the mutation test. In the latter case, the mutant program is equivalent to the original program.
Hong Zhu et al (1997) said that it requires huge human resources to determine if live mutants are equivalent to the original program.

Chen et al (1997) mentioned that if the literal $x_t$ evaluates to the same truth value 1 (0) for all Unique True Points in $\text{UTP}_i(S)$ where $x_t$ ($\overline{x_t}$) does not appear in $P_i$, we need not consider the Literal Insertion Fault for the literal $x_t$ ($\overline{x_t}$) that will be made on $P_i$ because $S$ and $I$ are equivalent. They also state two conditions under which a specification $S$ and implementation $I$ are equivalent.

i. For any $x_1 \in \text{UTP}_i(S)$ and for any $x_2 \in \text{NFP}$, $x_1$ and $x_2$ differ at some literal other than $x^i_j$ ($j^{th}$ literal in the $i^{th}$ term).

ii. The literal $x_t$ evaluates to 1 for all Unique True Points in $\text{UTP}_i(S)$.

Weyuker et al (1994) mentioned that each mutant represents a “buggy” version of the program and if the test set cannot distinguish the two versions (that is, both versions produce the same output for every input), then the inserted bug (mutant) would go undetected by the test set. If a large number of mutants are not distinguished by the test set, then many bugs would go undetected.

The faulty implementation evaluates to the same truth value as that of the original specification for all the combinations of the input. In such situations, the fault condition proposed by Kuhn (1999) is ineffective to detect the fault. A method has been proposed to detect the above fault. These sort of faults have emerged while testing for Literal Insertion Fault, Term Insertion Fault and Literal Reference Fault. Some of the examples in which Null Fault occur is explained below.
5.2.1 Null Fault in Literal Insertion Fault – An Example

For a realistic example, the following specification from Weyuker et al (1994) is considered.

\[ S = (abc(f(g + g(h+i))(en+d) + n(jk + j/i))) \]  
\[ (5.1) \]

The specification in DNF is given below.

\[ S = (a \land b \land c \land f \land i \land m \land n) \lor (a \land b \land c \land e \land f \land h \land n) \lor (a \land b \land c \land e \land f \land g \land n) \lor (a \land b \land c \land e \land f \land i \land n) \lor (a \land b \land c \land d \land f \land i) \lor (a \land b \land c \land d \land f \land n) \lor (a \land b \land c \land d \land f \land g) \lor (a \land b \land c \land j \land k \land n) \]  
\[ (5.2) \]

Let a literal say \( g \) be inserted in the term \( P_i = (a \land b \land c \land d \land f \land i) \) in \( S \) to get \( LIF \).

\[ LIF = (a \land b \land c \land j \land l \land m \land n) \lor (a \land b \land c \land e \land f \land h \land n) \lor (a \land b \land c \land e \land f \land g \land n) \lor (a \land b \land c \land e \land f \land i \land n) \lor (a \land b \land c \land d \land f \land i) \lor (a \land b \land c \land d \land f \land g) \lor (a \land b \land c \land d \land f \land n) \lor (a \land b \land c \land j \land k \land n) \]

\[ = (a \land b \land c \land j \land l \land m \land n) \lor (a \land b \land c \land e \land f \land h \land n) \lor (a \land b \land c \land e \land f \land g \land n) \lor (a \land b \land c \land e \land f \land i \land n) \lor (a \land b \land c \land d \land f \land n) \lor (a \land b \land c \land d \land f \land g) \lor (a \land b \land c \land d \land f \land h) \lor (a \land b \land c \land j \land k \land n) \]

\[ = (a \land b \land c \land j \land l \land m \land n) \lor (a \land b \land c \land e \land f \land n) \lor (a \land b \land c \land e \land f \land n) \lor (a \land b \land c \land d \land f \land i) \lor (a \land b \land c \land d \land f \land i) \lor (a \land b \land c \land d \land f \land h) \lor (a \land b \land c \land j \land k \land n) \]

\[ = (a \land b \land c \land j \land l \land m \land n) \lor (a \land b \land c \land e \land f \land n) \lor (a \land b \land c \land e \land f \land n) \lor (a \land b \land c \land d \land f \land n) \lor (a \land b \land c \land d \land f \land n) \lor (a \land b \land c \land j \land k \land n) \]

\[ = S \]
Hence, it is observed that $S = LIF$, although there exists a fault in the original specification. That is, both $S$ and $LIF$ evaluate to the same truth value for all the combinations of the input. Thus it is clear that $LIF$ in this case is another form of $S$, hence $SLIF$ is Null. That is, the fault will not be detected using the fault condition $SLIF$.

Whenever $SLIF$ is Null, the following hierarchy (Lau et al 2001) does not hold.

$$SLIF \Rightarrow STOF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$
$$SLIF \Rightarrow SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$

That is, the test cases for Term Omission Fault, Literal Reference Fault, Literal Negation Fault, Term Negation Fault and Expression Negation Fault cannot be derived from the test set of Literal Insertion Fault. A number of situations in which $SLIF$ is Null occurred during our analysis. In these situations, it is proposed to hypothesize the test set for the next higher fault class, that is, Term Omission Fault or Literal Reference Fault and use it to derive the test sets for Literal Negation Fault, Term Negation Fault and Expression Negation Fault. Thus, the hierarchy to be followed in this situation is given below.

$$STOF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$
$$SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$

### 5.2.2 Null Fault in Term Insertion Fault – An Example

For a realistic example, the following specification from Weyuker et al (1994) is considered.

$$S = (ac + bd)e(fg + fh)$$

(5.3)
The specification in DNF is given below.

\[ S = (a \land c \land e \land f \land g) \lor (b \land d \land e \land f \land g) \lor (a \land c \land e\neg f \land h) \lor (b \land d \land e\neg f \land h) \] (5.4)

Let \( P_t = (a \land c \land e \land h \land g) \), obtained by replacing the literal \( f \) by \( h \) in the term \( P_i = (a \land c \land e \land f \land g) \) be inserted in \( S \) to get \( TIF \).

\[ TIF = (a \land c \land e \land f \land g) \lor (b \land d \land e \land f \land g) \lor (a \land c \land e \neg f \land h) \lor (b \land d \land e \neg f \land h) \lor (a \land c \land e \land h \land g \land f) \lor (a \land c \land e \land h \land g \land \neg f) \]

\[ = (a \land c \land e \land f \land g) \lor (b \land d \land e \land f \land g) \lor (a \land c \land e \neg f \land h) \lor (b \land d \land e \neg f \land h) \lor (a \land c \land e \land h \land g \land f) \lor (a \land c \land e \land h \land g \land \neg f) \]

\[ = (a \land c \land e \land f \land g) \lor (a \land c \land e \land f \land g \land h) \lor (b \land d \land e \land f \land g) \lor (a \land c \land e \neg f \land h) \lor (a \land c \land e \neg f \land h \land g) \lor (b \land d \land e \neg f \land h) \lor (a \land c \land e \land h \land g \land f) \lor (a \land c \land e \land h \land g \land \neg f) \]

\[ = (a \land c \land e \land f \land g) \lor (a \land c \land e \land f \land g \land h) \lor (b \land d \land e \land f \land g) \lor (a \land c \land e \neg f \land h) \lor (b \land d \land e \neg f \land h) \]

\[ = S \]

Hence, we observe that the faulty specification \( TIF \) is another form of the original specification \( S \). Thus, we observe that the fault condition \( S_{TIF} \) is unable to detect the Term Insertion Fault. That is, the fault condition proposed by Kuhn (1999)
does not detect the fault when the faulty implementation is another form of the original specification.

Whenever $Stif$ is Null, the following extended hierarchies do not hold

$$Stif \Rightarrow Slof \Rightarrow Slnf \Rightarrow Stnf \Rightarrow Senv$$

That is, the test cases for Literal Omission Fault, Literal Reference Fault, Literal Negation Fault, Term Negation Fault and Expression Negation Fault cannot be derived from the test set of Term Insertion Fault. A number of situations in which $Stif$ is Null occurred during our analysis. In these situations, it is proposed to hypothesize the test set of the next higher fault model, that is, Literal Omission Fault or Literal Reference Fault and use it to derive the test sets for Literal Negation Fault, Term Negation Fault and Expression Negation Fault. Thus, the hierarchy to be followed in this situation is given below.

$$Slof \Rightarrow Slnf \Rightarrow Stnf \Rightarrow Senv$$

$$Slrf \Rightarrow Slnf \Rightarrow Stnf \Rightarrow Senv$$

5.2.3 Null Fault in Literal Reference Fault – An Example

The following boolean specification given by Chen et al (1997) is considered.

$$S = (\overline{a} \land b) \lor (b \land c) \lor (a \land \overline{b}) \lor (c \land d)$$

Let the literal $b$ in the term $(b \land c)$ be replaced by the literal $a$ to get LRF. The faulty specification thus obtained is given below.
\[
LRF = (\bar{a} \land b) \lor (a \land c) \lor (a \land \bar{b}) \lor (\bar{c} \land d)
\]

Hence, it is seen that the faulty specification \(LRF\) is another form of the original specification \(S\). Hence \(SLRF\) is Null. That is, the fault will not be detected using the fault condition \(SLRF\).

Whenever \(SLRF\) is Null, the following hierarchies do not hold.

\[
SLIF \Rightarrow SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF
\]
\[
STIF \Rightarrow SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF
\]

That is, the test cases for Literal Negation Fault, Term Negation Fault and Expression Negation Fault cannot be derived from the test set of Literal Reference Fault. A number of situations in which \(SLRF\) is Null occurred during our analysis. It is observed that whenever \(SLRF\) is Null, \(SLIF\) and \(STIF\) are also Null.

In this situation, it is proposed to hypothesize the test set for Literal Omission Fault or Term Omission Fault and use it to derive the test sets for Literal Negation Fault, Term Negation Fault and Expression Negation fault. Thus, the hierarchy to be followed in this situation is given below.
The hierarchical structure in this case is given below:

\[
\begin{align*}
S_{\text{enf}} & \Rightarrow S_{\text{lnf}} \Rightarrow S_{\text{tnf}} \Rightarrow S_{\text{enf}} \\
S_{\text{lof}} & \Rightarrow S_{\text{lnf}} \Rightarrow S_{\text{tnf}} \Rightarrow S_{\text{enf}}
\end{align*}
\]

Figure 5.1 Hierarchy when SLRF is Null

5.2.4 How Null Fault occurs?

Few examples of Null Fault are already explained above. Analysis was carried out in trying to understand the cause of Null Fault. It was observed that the Null Fault is caused due to the adjustment of true points between different terms in a specification.

Consider the boolean specification given by Chen et al (1997). The specification is given in Equation (5.5). Let the literal \( b \) in the second term be replaced by \( a \). The LRF expression is given below.

\[
LRF = (\overline{a} \land b) \lor (a \land c) \lor (a \land \overline{b}) \lor (\overline{c} \land d)
\] (5.6)
In the previous section, it was observed that the faulty specification so obtained is another form of the original specification. Hence, the fault condition given by Kuhn (1999) does not generate any test cases to detect the fault. The table below contains the input values for which each term of the original specification and the faulty specification evaluates to true.

**Table 5.1 True points of Equations (5.5) and (5.6)**

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>4, 5, 6, 7</td>
<td>6, 7, 14, 15</td>
<td>8, 9, 10, 11</td>
<td>1, 5, 9, 13</td>
<td>1, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15</td>
</tr>
<tr>
<td>$LRF$</td>
<td>4, 5, 6, 7</td>
<td>10, 11, 14, 15</td>
<td>8, 9, 10, 11</td>
<td>1, 5, 9, 13</td>
<td>1, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15</td>
</tr>
</tbody>
</table>

$$R = P_1 \lor P_2 \lor P_3 \lor P_4$$

Here, $R$ is the set of true points for which the specification and its faulty implementation evaluates to True. As seen from the above table, because of the Literal Reference Fault, the true points 6 and 7, which evaluates to true for the second term in the original specification does not evaluate to true in the second term of the faulty specification. The test case 6 corresponds to the input value $a = 0, b = 1, c = 1$ and $d = 0$. The test case 7 corresponds to the input value $a = 0, b = 1, c = 1, d = 1$. But, for these values, the first term of the faulty specification evaluates to true. Effectively, the faulty specification evaluates to true for these values.

Similarly, because of the Literal Reference Fault, the second term of the faulty specification evaluates to true for the input values 10 and 11 (decimal value). The second term of the original specification does not evaluate to true for these values. But, the third term of the original specification evaluates to true for these values. Effectively, the original specification evaluates to true for these values.
Hence, it is observed that the original specification and the faulty specification evaluate to the same truth value for all the possible input combinations. Hence, the fault cannot be detected by the fault condition proposed by Kuhn (1999). The faulty specification thus obtained is another form of the original specification. In the following section, a method to detect the Null Fault is proposed.

5.2.5 Detection of Null Fault

Stocks et al (1993) gave the concept of Weak Mutation Testing. Whereas mutation testing (strong) required the output between the original program and the mutated program to be different, weak mutation testing required the intermediate values computed by the mutated code fragment to be different. The concept of weak mutation testing can be taken to detect Null Fault.

Two possibilities may arise during the implementation of a boolean specification when $S \oplus S' = \text{Null}$.

i. The specification $S$ may be implemented correctly. This can be confirmed when $S(P_i) \oplus S'(P_i)$ does not generate any test case.

ii. The specification $S$ may be implemented in its equivalent form. This can be ascertained when $S(P_i) \oplus S'(P_i)$ generate some test cases.

Here, $i = 1, 2, \ldots, m$ and $m$ is the number of terms in the specification $S$. $S'$ represents the faulty implementation of the specification. Consider the example where $SLRF$ is null. Table 5.1 gives the true points for each term of the original specification and Literal Reference Fault. The following table gives the result of term-wise Exclusive OR.
### Table 5.2 Detection of Null Fault

<table>
<thead>
<tr>
<th>Expression</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S(P_1) \oplus LRF(P_1)$</td>
<td>Null</td>
</tr>
<tr>
<td>$S(P_2) \oplus LRF(P_2)$</td>
<td>6, 7, 10, 11</td>
</tr>
<tr>
<td>$S(P_3) \oplus LRF(P_3)$</td>
<td>Null</td>
</tr>
<tr>
<td>$S(P_4) \oplus LRF(P_4)$</td>
<td>Null</td>
</tr>
</tbody>
</table>

It is clear from the above table that the term wise Exclusive OR generates some test cases when $S \oplus S'$ is Null. Thus, the Null Fault is detected.

### 5.3 EXCEPTIONS IN THE HIERARCHY OF FAULT CLASSES

While analyzing the hierarchy of fault classes, it is observed that in certain situations, the hierarchy proposed by Lau et al (2001) and Tsuchiya et al (2002) does not hold. This is because one of the intermediate fault classes is either empty due to the Null Fault or it is disjoint with other fault classes in the hierarchy. Other exceptions such as $SLRF \Rightarrow SLOF$, $SLOF \Rightarrow SLRF$ and $SLOF = SLRF$ are also discussed.

The situations discussed herewith are not exhaustive and further study is possible. The examples presented are formal specifications used in Traffic Alert and Collision Avoidance Subsystem (TCAS II) given by Weyuker et al (1994).

#### 5.3.1 $SLOF$ and $SLRF$ are disjoint

Kuhn (1999) showed that there is a hierarchy of fault classes with respect to detection capability. More specifically, he showed that test cases that detect Variable Negation Fault also detect Expression Negation Fault and test cases that detect Variable Reference Faults also detect Variable Negation Fault. He also proposed a new fault class namely, Missing Condition Fault. He mentioned that the
faults in this class should be hypothesized to generate efficient tests, since Missing Condition Fault is often equivalent to Variable Reference Fault, thus leading to higher detection capability.

Tsuchiya et al (2002) mentioned that the conclusion of Kuhn (1999) was premature since the relationship between Missing Condition Fault and faults in other classes had not been sufficiently investigated. The above situation is taken for analysis for the fault model proposed by Lau et al (2001). The fault class \( TNF \) is same as \( ENF \) proposed by Kuhn (1999).


In most of the situations, the common point between Literal Reference Fault and Literal Omission Fault will detect the four faults as mentioned by Tsuchiya et al (2002). During the analysis of test sets for various faults, in certain situations it was observed that the test set for Literal Reference Fault and Literal Omission Fault were disjoint. One such situation is explained below.

Consider the TCAS II specification given in Equation (5.4). Let the literal \( f \) be omitted in the term \( P_i = (b \land d \land e \land f \land g) \) in \( S \) to get \( LOF \). Let the same literal \( f \) be replaced by another literal \( h \) in the same term \( P_i \) to get \( LRF \). The test set for \( SLOF \) and \( SLRF \) is given below.
Table 5.3 Disjoint test set of $S_{LOF}$ and $S_{LRF}$

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{LOF}$</td>
<td>90, 122, 218, 250</td>
</tr>
<tr>
<td>$S_{LRF}$</td>
<td>94, 126, 222</td>
</tr>
</tbody>
</table>

A number of situations were encountered in which the test set for Literal Omission Fault and Literal Reference Fault were disjoint. Hence, there was no point common in the two test sets. Hence, the following hierarchy stated by Tsuchiya et al (2002) does not hold in certain situations.

$S_{LOF} \land S_{LRF} \Rightarrow S_{LNF} \Rightarrow S_{ENF}$

Hence, in these situations it is proposed to derive the test set for Literal Reference Fault and Literal Omission Fault separately and use it to derive the test set for Literal Negation Fault and Expression Negation Fault. That is, the hierarchy to be followed in this situation is as follows:

$S_{LOF} \Rightarrow S_{LNF} \Rightarrow S_{ENF}$

$S_{LRF} \Rightarrow S_{LNF} \Rightarrow S_{ENF}$

5.3.2 $S_{LOF}$ implies $S_{LRF}$

In most cases, some points from the test set of Literal Omission Fault are present in the test set of Literal Reference Fault. In some situations, all the points present in the test set of Literal Omission Fault are present in the test set of Literal Reference Fault. One such situation is given below:

Consider the TCAS II specification given in Equation (5.2). Let the literal $\bar{j}$ be omitted in the term $P_j = (a \land b \land c \land \bar{j} \land l \land \bar{m} \land \bar{n})$ in $S$ to get $LOF$. Let the same
literal \( j \) be replaced by another literal \( k \) in the same term \( P \) to get \( LRF \). The test sets for \( SLOF \) and \( SLRF \) is given below.

Table 5.4 Test set of \( SLOF \) and \( SLRF \) when \( SLOF \Rightarrow SLRF \)

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SLOF )</td>
<td>8212, 8244, 8276, 8308, 8340, 8372, 8404, ...</td>
</tr>
<tr>
<td>( SLRF )</td>
<td>8204, 8212, 8244, 8276, 8308, 8340, 8372, 8404, ...</td>
</tr>
</tbody>
</table>

In this situation, the following hierarchy (Lau et al 2001)

\[ SLOF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF \]

\[ SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF \]

will be as follows:

\[ SLOF \Rightarrow SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF \]

5.3.3 \( SLRF \) implies \( SLOF \)

Generally, some test cases from the test set of Literal Reference Fault exist in the test set of the corresponding Literal Omission Fault. In certain special situations, all the test cases in the test set of Literal Reference Fault are present in the test set of Literal Omission Fault. One such situation is explained here. Consider the following TCAS II specification.

\[ S = ((ac + bd)e(f + (i(gj + hk)))) \]

The specification in DNF is given below.
Let the literal $b$ be omitted in the term \( P_j = (b \land d \land e \land g \land i \land j) \) in $S$ to get $LOF$. Let the same literal $b$ be replaced by another literal \( f \) in the same term $P_j$ to get $LRF$. The test sets for $SLOF$ and $SLRF$ is given below.

### Table 5.5 Test set of $SLOF$ and $SLRF$ when $SLRF \Rightarrow SLOF$

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SLRF$</td>
<td>214, 215, 222, 223, 470, 471, 478, 479, 1238, 1239, 1246, 1247</td>
</tr>
<tr>
<td>$SLOF$</td>
<td>214, 215, 222, 223, 246, 247, ..., 470, 471, 478, 479, ..., 1238, 1239, 1246, 1247, ...</td>
</tr>
</tbody>
</table>

In this situation, the following hierarchy (Lau et al 2001)

$$SLOF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$

$$SLRF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$

will be as follows:

$$SLRF \Rightarrow SLOF \Rightarrow SLNF \Rightarrow STNF \Rightarrow SENF$$

### 5.3.4 $SLOF$ equal to $SLRF$

Generally, there are some test cases common to the test sets of Literal Omission Fault and Literal Reference Fault. In certain situations, the two test sets are the same. That is, both the test sets contain the same set of test cases. One such situation is explained below.
Consider the TCAS II specification given in Equation (5.2). Let the literal \( i \) be omitted in the term \( P_i = (a \land b \land c \land d \land f \land i) \) in \( S \) to get \( \text{LOF} \). Let the same literal \( i \) be replaced by another literal \( g \) in the same term \( P_i \) to get \( \text{LRF} \). The test sets for \( \text{SLOF} \) and \( \text{SLRF} \) are given below.

### Table 5.6 Test set of \( \text{SLOF} \) and \( \text{SLRF} \) when \( \text{SLOF} = \text{SLRF} \)

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{SLOF} )</td>
<td>9472, 9473, 9474, 9475, 9477, 9478, ...</td>
</tr>
<tr>
<td>( \text{SLRF} )</td>
<td>9472, 9473, 9474, 9475, 9477, 9478, ...</td>
</tr>
</tbody>
</table>

In this situation, the following hierarchy (Lau et al 2001):

\[
\text{SLOF} \Rightarrow \text{SLNF} \Rightarrow \text{STNF} \Rightarrow \text{SENF}
\]

\[
\text{SLRF} \Rightarrow \text{SLNF} \Rightarrow \text{STNF} \Rightarrow \text{SENF}
\]

will be as follows:

\[
\text{SLRF} = \text{SLOF} \Rightarrow \text{SLNF} \Rightarrow \text{STNF} \Rightarrow \text{SENF}
\]

### 5.4 RESULTS

During our experimental analysis, it was observed that by formulating expressions, it is possible to derive test sets for Literal Insertion Fault, Term Insertion Fault, Term Omission Fault, Literal Reference Fault, Literal Omission Fault, Term Negation Fault and Expression Negation Fault from Literal Negation Fault. The results are given below.
Theorem 5.1: If the literal negated in the term $P_i$ of Literal Negation Fault is the same term in which a literal $x_t$ is inserted for Literal Insertion Fault, then a test case $x$ will detect a Literal Insertion Fault if and only if $x \in (S \land LNF \land x_t)$.

Proof: Let $S$ be a boolean specification in Disjunctive Normal Form. The expression for $S$ is given below.

$$S = (a_1 \land a_2 \land \ldots \land a_k) \lor (b_1 \land b_2 \land \ldots \land b_l) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y) \quad (5.9)$$

Let a literal say $b_i$ be inserted in the term $P_1$ in $S$ to get $LIF$. Let the literal $a_i$ in the term $P_1$ in $S$ be negated to get $LNF$. The expressions for $LNF$ and $LIF$ are given below.

$$LNF = (\overline{a_1} \land a_2 \land \ldots \land a_k) \lor (b_1 \land b_2 \land \ldots \land b_l) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y) \quad (5.10)$$

$$LIF = (b_1 \land a_1 \land a_2 \land \ldots \land a_k) \lor (b_1 \land b_2 \land \ldots \land b_l) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y) \quad (5.11)$$

The fault condition for Literal Insertion Fault, computed using Kuhn’s strategy is given below.

$$SLIF = S \oplus LIF$$

$$= ((a_1 \land a_2 \land \ldots \land a_k) \lor (b_1 \land b_2 \land \ldots \land b_l) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y)) \oplus ((b_1 \land a_1 \land a_2 \land \ldots \land a_k) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y))$$

$$= ((a_1 \land \overline{b_i}) \land a_2 \land \ldots \land a_k) \land (\overline{b_1} \lor \overline{b_2} \lor \ldots \lor \overline{b_l}) \land \ldots \land (\overline{z_1} \lor \overline{z_2} \lor \ldots \lor \overline{z_y}) \quad (5.12)$$

Let $x$ be a test case such that $x \in (S \land LNF \land \overline{x_t})$. Here $x_t = b_i$. The expression $(S \land LNF \land \overline{b_i})$ is evaluated below.
It is clear from Equation (5.12) and Equation (5.13) that \( S \land \overline{LNF} \land \overline{b_1} \) = \( S \cup LNF \land \overline{x_1} \). That is, if the literal inserted in a term in \( S \) is the same term in which a Literal Negation Fault occurs, then a test case \( x \) will detect a Literal Insertion Fault if and only if \( x \in (S \land \overline{LNF} \land \overline{x_1}) \).

**Theorem 5.2:** If the term inserted in Term Insertion Fault is the same term in which a Literal Negation Fault occurs, then a test case will detect a Term Insertion Fault if and only if \( x \in (S \land \overline{LNF} \land \overline{x_1}) \).

**Proof of Theorem 5.2:** Let \( S \) be a boolean specification in Disjunctive Normal Form as given in Equation (5.9). Let the term inserted in \( S \) to get TIF be \( P_t = (b_1 \land a_2 \land \ldots \land a_k) \). The term \( P_t \) is obtained by replacing the literal \( a_1 \) by the literal \( b_1 \) in the term \( P_1 = (a_1 \land a_2 \land \ldots \land a_k) \) in \( S \). Let the same literal \( a_1 \) be negated to get Literal Negation Fault. The expression for TIF is given below.

\[
TIF = (b_1 \land a_2 \land \ldots \land a_k) \lor (a_1 \land a_2 \land \ldots \land a_k) \\
\lor (b_1 \land b_2 \land \ldots \land b_k) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y) \quad (5.14)
\]
The fault condition for Term Insertion Fault, computed using Kuhn's strategy is given below.

\[ S_{TIF} = S \oplus TIF \]
\[ = ((a_1 \lor a_2 \land \ldots \land a_k) \lor (b_1 \lor b_2 \land \ldots \land b_k)) \lor \ldots \lor (z_1 \lor z_2 \land \ldots \lor z_p) \]
\[ ((b_1 \lor a_2 \land \ldots \land a_k) \lor (a_1 \lor a_2 \land \ldots \land a_k) \lor (b_1 \lor b_2 \land \ldots \lor b_k) \]
\[ \lor \ldots \lor (z_1 \lor z_2 \land \ldots \lor z_p) \]
\[ = ((\bar{a}_1 \lor \bar{a}_2) \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor \bar{b}_2) \lor \ldots \lor (\bar{z}_1 \lor \bar{z}_2) \land \ldots \land (\bar{z}_1 \lor \bar{z}_2) \lor \ldots \lor (\bar{z}_1 \lor \bar{z}_2) \] (5.15)

Let \( x \) be a test case such that \( x \in (S \land LNF \land x_t) \). Here \( x_t = b_l \). The expression \( \bar{S} \land LNF \land b_l \) is evaluated below.

\[ \bar{S} \land LNF \land b_l = ((a_1 \land a_2 \land \ldots \land a_k) \lor (b_1 \land b_2 \land \ldots \land b_k) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_p)) \land \]
\[ ((\bar{a}_1 \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor \bar{b}_2) \lor \ldots \lor (\bar{z}_1 \lor \bar{z}_2) \land \ldots \land (\bar{z}_1 \lor \bar{z}_2) \land b_l \]
\[ = ((\bar{a}_1 \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor \bar{b}_2) \lor \ldots \lor (\bar{z}_1 \lor \bar{z}_2) \land \ldots \land (\bar{z}_1 \lor \bar{z}_2) \land b_l) \] (5.16)

It is clear from Equation (5.15) and Equation (5.16) that \( (\bar{S} \land LNF \land x_t) = S_{TIF} \). That is, if the literal negated in Literal Negation Fault is the same literal replaced in the same term that is inserted in Term Insertion Fault, then a test case \( x \) will detect a Term Insertion Fault if and only if \( x \in (\bar{S} \land LNF \land x_t) \).

**Theorem 5.3:** If the literal negated in Literal Negation Fault is the same literal omitted in Literal Omission Fault, then a test case \( x \) will detect a Literal Omission Fault if and only if \( x \in (\bar{S} \land LNF) \).
Proof: Let $S$ be a boolean specification in Disjunctive Normal Form. Let a literal say $a_j$ be negated in the term $P_i$ in $S$ to get Literal Negation Fault. Let the same literal $a_j$ be omitted in $P_i$ in $S$ to get $LOF$. The expressions for $S$ and $LNF$ are given in Equation (5.9) and Equation (5.10) respectively. The expression for $LOF$ is given below.

$$LOF = (a_3 \land \ldots \land a_{aj} \lor b_1 \lor b_2 \lor \ldots \land b_{aj} \lor \ldots \lor z_1 \lor z_2 \land \ldots \land z_y) \quad (5.17)$$

The fault condition for Literal Omission Fault, computed by Kuhn's strategy is given below.

$$SLOF = S \oplus LOF$$

$$= (((a_1, a_2 \land \ldots \land a_{aj}) \lor (b_1 \lor b_2 \land \ldots \land b_{aj}) \lor \ldots \lor (z_1 \lor z_2 \land \ldots \land z_y)) \oplus$$

$$\quad (((a_2 \land \ldots \land a_{aj}) \lor (b_1 \lor b_2 \land \ldots \land b_{aj}) \lor \ldots \lor (z_1 \lor z_2 \land \ldots \land z_y))$$

$$= (\overline{a_1}, a_2 \land \ldots \land a_{aj}) \land (\overline{b_1} \lor \overline{b_2} \lor \ldots \lor \overline{b_{aj}}) \land \ldots \land (\overline{z_1} \lor \overline{z_2} \lor \ldots \lor \overline{z_y}) \quad (5.18)$$

Let $x$ be a test case such that $x \in (\overline{S} \land LNF)$. The expression for $\overline{S} \land LNF$ is given below.

$$\overline{S} \land LNF = (((a_1 \land a_2 \land \ldots \land a_{aj}) \lor (b_1 \lor b_2 \land \ldots \land b_{aj}) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y)) \land$$

$$\quad (((\overline{a_1} \land a_2 \land \ldots \land a_{aj}) \lor (\overline{b_1} \lor b_2 \land \ldots \land b_{aj}) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_y))$$

$$= (\overline{a_1}, a_2 \land \ldots \land a_{aj}) \land (\overline{b_1} \lor \overline{b_2} \lor \ldots \lor \overline{b_{aj}}) \land \ldots \land (\overline{z_1} \lor \overline{z_2} \lor \ldots \lor \overline{z_y}) \quad (5.19)$$

It is clear from Equation (5.18) and Equation (5.19) that $\overline{S} \land LNF = SLOF$. That is, if the literal negated in Literal Negation Fault is the same literal omitted in Literal Omission Fault, then a test case $x$ will detect a Literal Omission Fault if and only if $x \in (\overline{S} \land LNF)$. 


**Theorem 5.4:** If the term \( P_i \) in which a literal is negated to get Literal Negation Fault is the same term omitted in Term Omission Fault, then a test case \( x \) will detect a Term Omission Fault if and only if \( x \in (S \land \text{LNF}) \).

**Proof:** Let \( S \) be a boolean specification in Disjunctive Normal Form as given in Equation (5.9). Let the term \( P_i \) in \( S \) be omitted to get Term Omission Fault. Let a literal say \( a_1 \) in \( P_i \) in \( S \) be negated to get Literal Negation Fault. The expression for \( \text{LNF} \) is given in Equation (5.10) and the expression for \( \text{TOF} \) is given below.

\[
\text{TOF} = (b_1 \land b_2 \land \ldots \land b_J \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_J)) \tag{5.20}
\]

The fault condition for Term Omission Fault, computed by Kuhn's strategy is given below.

\[
\text{Stof} = S \oplus \text{TOF} \\
= \left( (a_1 \land a_2 \land \ldots \land a_J) \lor (b_1 \land b_2 \land \ldots \land b_J) \right) \lor \ldots \lor \left( (z_1 \land z_2 \land \ldots \land z_J) \right) \\
= (a_1 \land a_2 \land \ldots \land a_J) \land (b_1 \lor \ldots \lor b_J) \land \ldots \land (z_1 \lor \ldots \lor z_J) \tag{5.21}
\]

Let \( x \) be a test case such that \( x \in (S \land \text{LNF}) \). The expression for \( S \land \text{LNF} \) is given below.

\[
S \land \text{LNF} = \left( (a_1 \land a_2 \land \ldots \land a_J) \lor (b_1 \land b_2 \land \ldots \land b_J) \right) \land \ldots \land \left( (z_1 \land z_2 \land \ldots \land z_J) \right) \\
= (a_1 \land a_2 \land \ldots \land a_J) \land (b_1 \lor \ldots \lor b_J) \land \ldots \land (z_1 \lor \ldots \lor z_J) \tag{5.22}
\]

It is clear from Equation (5.21) and Equation (5.22) that \( (S \land \text{LNF}) = \text{Stof} \). That is, if the term in which a literal is negated to get Literal
Negation Fault is the same term omitted in Term Omission Fault, then a test case $x$ will detect a Term Omission Fault if and only if $x \in (S \land \overline{LNF})$.

**Theorem 5.5:** If the literal negated in Literal Negation Fault is the same literal replaced in Literal Reference Fault, then a test case $x$ will detect a Literal Reference Fault if and only if $x \in (((S \land \overline{LNF} \land x_t) \lor (S \land \overline{LNF} \lor x_t))$.

**Proof:** Let $S$ be a boolean specification in Disjunctive Normal Form as given in Equation (5.9). Let the literal $a_1$ in the term $P_1$ in $S$ be replaced by the literal $b_1$ in the term $P_2$ in $S$ to get Literal Reference Fault. Let the same literal $a_1$ in the same term $P_1$ be negated to get Literal Negation Fault. The expression for $LNF$ is given in Equation (5.10) and for $LRF$ is given below.

$$LRF = (b_1 \land a_2 \land ... \land a_k) \lor (b_1 \land b_2 \land ... \land b_n) \lor ... \lor (z_1 \land z_2 \land ... \land z_y) \quad (5.23)$$

The fault condition for Literal Reference Fault, computed by Kuhn's strategy is given below.

$$S_{LRF} = S \oplus LRF$$

$$= (((a_1 \land a_2 \land ... \land a_k) \lor (b_1 \land b_2 \land ... \land b_n) \lor ... \lor (z_1 \land z_2 \land ... \land z_y)) \oplus$$

$$((b_1 \land a_2 \land ... \land a_k) \lor (b_1 \land b_2 \land ... \land b_n) \lor ... \lor (z_1 \land z_2 \land ... \land z_y))$$

$$= (((a_1 \land \overline{b}_1) \lor (\overline{a}_1 \land b_1)) \land a_2 \land ... \land a_k) \land (\overline{b}_1 \lor b_2 \lor ... \lor b_n)$$

$$\land ... \land (\overline{z}_1 \lor \overline{z}_2 \lor ... \lor \overline{z}_y) \quad (5.24)$$

Let $x$ be a test case such that $x \in (S \land \overline{LNF} \land x_t)$ or $x \in (S \land LNF \land x_t)$. Here $x_t = b_1$. The expression $((S \land \overline{LNF} \land \overline{b}_1) \lor (S \land LNF \land b_1))$ is given below (Refer Equations (5.13) and Equation (5.16)).
It is clear from Equation (5.24) and Equation (5.25) that 
\[ ((S \land LNF \land \bar{b}_i) \lor (\bar{S} \land LNF \land \bar{b}_i)) \]
\[ = (((a_1 \land \bar{b}_i) \land a_2 \land \ldots \land a_k) \land \bar{b}_1 \land v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k)) \]
\[ = (((\bar{a}_1 \land a_2 \land \ldots \land a_k) \land \bar{b}_1 \land v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k)) \]
\[ = (((a_1 \land \bar{b}_i) \lor (\bar{a}_1 \land b_i)) \land a_2 \land \ldots \land a_k) \land \bar{b}_1 \land v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k) \]
\[ = ((a_1 \land \bar{b}_i) \lor (\bar{a}_1 \land b_i)) \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k) \]
\[ = (((a_1 \land \bar{b}_i) \lor (\bar{a}_1 \land b_i)) \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k) \]
\[ = (((a_1 \land \bar{b}_i) \lor (\bar{a}_1 \land b_i)) \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k) \]
\[ = ((a_1 \land \bar{b}_i) \lor (\bar{a}_1 \land b_i)) \land a_2 \land \ldots \land a_k) \land (\bar{b}_1 \lor v \bar{b}_2 \land \ldots \land v \bar{b}_k) \land \ldots \land \bar{v} (\bar{z}_1 \lor \bar{z}_2 \lor \ldots \lor \bar{z}_k) \]

### Relation between $S_{LNF}$ and $S_{TNF}$

The relationship between $S_{LNF}$ and $S_{TNF}$ is $S_{LNF} \Rightarrow S_{TNF}$. The proof is given in Theorem 4.14 in Chapter 4.

### Theorem 5.6

If the specification in which a literal in a term is negated to get Literal Negation Fault is the same specification that is negated to get Expression Negation Fault, then a test case that detects Literal Negation Fault will also detect Expression Negation Fault.

#### Proof:

Let $S$ be a boolean specification in Disjunctive Normal Form as given in Equation (5.9). Let the literal $a_1$ in $P_1$ in $S$ be negated to get Literal Negation Fault. Let the same specification $S$ be negated to get Expression Negation Fault. The expression for $LNF$ is given in Equation (5.10) and the expression for $ENF$ is given below.

\[ ENF = ((a_1 \land a_2 \land \ldots \land a_k) \lor (b_1 \land b_2 \land \ldots \land b_s) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_r)) \]  (5.26)
The fault condition for Literal Negation Fault, computed by Kuhn's strategy is given below.

\[ S_{lnf} = S \oplus LNF \]
\[ = ((a_1 \land a_2 \land \ldots \land a_l) \lor (b_1 \land b_2 \land \ldots \land b_m) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_p)) \oplus \]
\[ (a_1, a_2, \ldots, a_l, b_1, b_2, \ldots, b_m, \ldots, z_1, z_2, \ldots, z_p) \]
\[ = (a_1 \land \ldots \land a_l) \land (b_1 \lor \ldots \lor b_m) \land \ldots \land (z_1 \lor \ldots \lor z_p) \] (5.27)

The fault condition for Expression Negation Fault, computed by Kuhn's strategy is given below.

\[ S_{enf} = S \oplus ENF \]
\[ = ((a_1 \land a_2 \land \ldots \land a_l) \lor (b_1 \land b_2 \land \ldots \land b_m) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_p)) \oplus \]
\[ (a_1 \land a_2 \land \ldots \land a_l) \lor (b_1 \land b_2 \land \ldots \land b_m) \lor \ldots \lor (z_1 \land z_2 \land \ldots \land z_p) \]
\[ = T \]

That is, Expression Negation Fault can be detected using any input condition. That is, if a test case detects Literal Negation Fault, it would also detect Expression Negation Fault if the literal negated in Literal Negation Fault is from the same specification that is negated to get Expression Negation Fault.

5.5 NEED TO GO FOR LITERAL NEGATION FAULT

Here, we discuss some of the situations that make it important to consider Literal Negation Fault for deriving test cases for various fault classes even though Literal Reference Fault can do the same task.

The only major difference between deriving test cases using Literal Reference Fault and Literal Negation Fault is that in using LRF, we are able to derive
only a minimal test set for Literal Omission Fault and Term Omission Fault whereas using $LNF$, we derive the complete test set for Literal Omission Fault and Term Omission Fault. Test set generation from Literal Negation Fault is important in situations in which the hierarchy of fault classes does not help to generate the test cases for various faults. The cases are discussed with an example below.

**Case 1: SLOF and SLRF are disjoint**

In this case, it is not possible to derive the test sets for Literal Negation Fault and Term Negation Fault as given by Tsuchiya et al (2002). To derive the test set for various faults, Literal Negation Fault is used.

Consider the TCAS II specification given in Equation (5.4). Let the literal $f$ be omitted in the term $P_i = (b \land d \land e \land f \land g)$ in $S$ to get LOF. Let the same literal $f$ be replaced by another literal $h$ in the same term $P_i$ to get LRF. Let the literal $f$ from the same term $P_i$ in $S$ be negated to get LNF. The test sets for Literal Reference Fault and Literal Omission Fault are given in the following table.

**Table 5.7 Test set of SLOF and SLRF from LNF when SLOF and SLRF are disjoint**

<table>
<thead>
<tr>
<th>Fault condition / Test Expression</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SLRF$</td>
<td>94, 126, 222</td>
</tr>
<tr>
<td>$SLOF$</td>
<td>90, 122, 218, 250</td>
</tr>
<tr>
<td>$((S \land \overline{LNF} \land h) \lor (\overline{S} \land LNF \land h)) = SLRF$</td>
<td>94, 126, 222</td>
</tr>
<tr>
<td>$S \land LNF = SLOF$</td>
<td>90, 122, 218, 250</td>
</tr>
</tbody>
</table>

The test sets generated for various faults from Literal Negation Fault were verified by generating test sets for the corresponding faults by Kuhn’s strategy and they were found to be right. Hence, it is clear from the above table that whenever $SLOF$ and $SLRF$ are disjoint, the hierarchy of fault classes proposed by Tsuchiya et al (2002)
cannot be followed to generate test cases for various faults and Literal Negation Fault can be used to generate test cases for various faults.

Case 2: Null Fault occurs in Literal Reference Fault

In this case, it is not possible to derive the test set for Literal Negation Fault, Term Negation Fault and Expression Negation Fault as shown in Chapter 4. To derive the test set for various faults, Literal Negation Fault is used.

Consider the TCAS II specification given in Equation (5.5). Let a literal say \( b \) be replaced by the literal \( a \) in the term \( P_2 = (b \land c) \) in \( S \) to get \( LRF \). Let the literal \( b \) from the same term \( P_2 \) in \( S \) be negated to get \( LNF \). The test set for all the eight faults is given in the following table:

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SLRF )</td>
<td>Null</td>
</tr>
<tr>
<td>( SLIF )</td>
<td>Null</td>
</tr>
<tr>
<td>( STIF )</td>
<td>Null</td>
</tr>
<tr>
<td>( S \land LNF = SLOF )</td>
<td>2, 3</td>
</tr>
<tr>
<td>( S \land LNF = STOF )</td>
<td>14, 15</td>
</tr>
<tr>
<td>( SLNF )</td>
<td>2, 3, 14, 15</td>
</tr>
<tr>
<td>( STNF )</td>
<td>0; 2, 3, 12, 14, 15</td>
</tr>
<tr>
<td>( SENF )</td>
<td>0, 1, 2, 3, ..., 12, 13, 14, 15</td>
</tr>
</tbody>
</table>

The test sets generated for various faults from Literal Negation Fault were verified by generating test sets for the corresponding faults by Kuhn's strategy and they were found to be right. Hence, it is clear from the above table that whenever \( SLRF \) is Null, the hierarchy of fault classes cannot be followed to generate test cases for various faults and Literal Negation Fault can be used to generate test cases for various faults.
Case 3: Null Fault occurs in Literal Insertion Fault

In this case, it is not possible to derive the test for Term Omission Fault, Literal Reference Fault, Literal Negation Fault, Term Negation Fault and Expression Negation Fault as discussed in Chapter 4. To derive the test set for these faults, Literal Negation Fault is used.

Consider the TCAS II specification given in Equation (5.2). Let a literal say $a$ be replaced by the literal $\overline{a}$ in the term $P_i = (a \land \overline{b} \land \overline{c} \land \overline{d} \land \overline{f} \land \overline{g} \land \overline{n})$ to get $LRF$. Let the literal $\overline{a}$ be inserted in the term $P_i$ to get $LIF$. Let the same term $P_i$ be omitted to get $TOF$. Let a literal say $a$ in the same term $P_i$ be negated to get $LNF$. The test sets for $SLIF$, $STOF$, $SLRF$, $SLNF$, $STNF$ and $SSENF$ from Literal Negation Fault are given in the following table.

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SLIF$</td>
<td>Null</td>
</tr>
<tr>
<td>$S \land LNF = STOF$</td>
<td>8576, 8578, 8582, ...</td>
</tr>
<tr>
<td>$((S \land LNF \land \overline{a}) \lor (\overline{S} \land LNF \land \overline{d})) = SLRF$</td>
<td>384, 386, 388, 390, 392, ...</td>
</tr>
<tr>
<td>$SLNF$</td>
<td>384, 386, 388, 390, 392, ..., 1408, ..., 8576, 8578, 8582, 8584, ...</td>
</tr>
<tr>
<td>$STNF$</td>
<td>0, 1, ..., 384, 386, 388, 390, 392, ..., 1408, ..., 8576, 8578, 8582, 8584, ...</td>
</tr>
<tr>
<td>$SSENF$</td>
<td>0, 1, ..., 384, 385, 386, 387, 388, 389, 390, 391, 392, ..., 1408, ..., 8576, 8577, 8578, 8579, 8580, 8581, 8582, 8583, 8584, ..., 8196, ...</td>
</tr>
</tbody>
</table>

The test sets generated for various faults from Literal Negation Fault were verified by generating test sets for corresponding faults by Kuhn's strategy and they were found to be right. Hence, it is clear from the above table that whenever $SLIF$ is Null, the hierarchy of fault classes cannot be followed to generate test cases for various faults and Literal Negation Fault can be used to generate test cases for various faults.
Case 4: Null Fault occurs in Term Insertion Fault

In this case, it is not possible to derive the test set for Literal Omission Fault, Literal Reference Fault, Literal Negation Fault, Term Negation Fault and Expression Negation Fault as stated in Chapter 4. To derive the test set for various faults, Literal Negation Fault is used.

Consider the TCAS II specification given in Equation (5.8). Let a literal say $g$ be replaced by the literal $f$ in the term $P_i = (a \land c \land e \land g \land i \land j)$ in $S$ to get $LRF$. Let the literal $g$ from the same term $P_i$ in $S$ be omitted to get $LOF$. Let the same literal $g$ be negated in the same term $P_i$ in $S$ to get $LNF$. The test sets for $STIF$, $\overline{S} \land LNF$, $SLRF$, $SLNF$, $STNF$ and $SENF$ from Literal Negation Fault is given below:

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$STIF$</td>
<td>Null</td>
</tr>
<tr>
<td>$\overline{S} \land LNF = SLOF$</td>
<td>$1350, 1351, 1358, ...$</td>
</tr>
<tr>
<td>$((S \land LNF \land f) \lor (\overline{S} \land LNF \land f)) = SLOF$</td>
<td>$1366, 1367, 1374, ...$</td>
</tr>
<tr>
<td>$SLNF$</td>
<td>$1350, 1351, 1358, 1366, 1367, 1374, ...$, $1886, ...$</td>
</tr>
<tr>
<td>$STNF$</td>
<td>$0, 1, ...$, $1350, 1351, 1358, 1366, 1367, 1374, ...$, $1886, ...$</td>
</tr>
<tr>
<td>$SENF$</td>
<td>$0, 1, ...$, $717, 718, 719, ...$, $1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, ...$, $1366, 1367, ...$, $1374, ...$, $1886, ...$</td>
</tr>
</tbody>
</table>

The test sets generated for various faults from Literal Negation Fault were verified by generating test sets for corresponding faults by Kuhn’s strategy and they were found to be right. Hence, it is clear from the above table that whenever $STIF$ is Null, the hierarchy of fault classes cannot be followed to generate test cases for various faults and Literal Negation Fault can be used to generate test cases for various faults.
5.6 APPLICATIONS

The theorems stated in the chapter were verified using formal specifications used in TCAS II.

Example for Theorem 5.1: Consider the following TCAS II specification given in Equation (5.8). Let a literal say $\text{a}$ be inserted in the term $P* = (b \land d \land e \land f)$ in $S$ to get $\text{LIF}$. Let a literal say $e$ in the same term $P_t$ be negated to get $\text{LNF}$. The table below shows the test set for $SLIF$ and $S \land LNF \land \overline{a}$.

Table 5.11 Test set of $SLIF$ from Literal Negation Fault

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \land LNF \land a$</td>
<td>736, 737, 738, 739, 740, 741, 742, 743, ...</td>
</tr>
<tr>
<td>$SLIF$</td>
<td>736, 737, 738, 739, 740, 741, 742, 743, ...</td>
</tr>
</tbody>
</table>

From the table it is clear that $S \land LNF \land \overline{x_t} = SLIF$.

Example for Theorem 5.2: Consider the following TCAS II specification given in Equation (5.4). Let the term $P_t = (a \land d \land e \land f \land g)$, obtained by replacing the literal $b$ by $a$ in the term $P_t = (b \land d \land e \land f \land g)$ in $S$ to get $\text{TIF}$. Let the same literal $b$ in the same term $P_t$ be negated to get $\text{LNF}$. The table below shows the test set for $STIF$ and $S' \land LNF \land a$.

Table 5.12 Test set of $STIF$ from Literal Negation Fault

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S' \land LNF \land a$</td>
<td>158, 159</td>
</tr>
<tr>
<td>$STIF$</td>
<td>158, 159</td>
</tr>
</tbody>
</table>

From the table it is clear that $S' \land LNF \land x_t = STIF$. 
Example for Theorem 5.3: Consider the TCAS II specification given in Equation (5.4). Let a literal say $h$ in the term $P_i = (a \land c \land e \land \overline{f} \land h)$ in $S$ be omitted to get $LOF$. Let the same literal $h$ in the same term $P_i$ be negated to get $LNF$. The table below shows the test set for $SLOF$ and $\overline{S} \land LNF$.

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{S} \land LNF$</td>
<td>168, 170, 184, 186, 232, 234, 248, 250</td>
</tr>
<tr>
<td>$SLOF$</td>
<td>168, 170, 184, 186, 232, 234, 248, 250</td>
</tr>
</tbody>
</table>

From the table it is clear that $\overline{S} \land LNF = SLOF$.

Example for Theorem 5.4: Consider the following TCAS II specification given in Equation (5.2). Let the term $P_i = (a \land b \land c \land d \land f \land h)$ in $S$ be omitted to get $TOF$. Let a literal say $a$ in the same term $P_i$ be negated to get $LNF$. The table below shows the test set for $STOF$ and $S \land LNF$.

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \land LNF$</td>
<td>9537, 9539, 9541, 9543, 9545, 9547, 9549, ...</td>
</tr>
<tr>
<td>$STOF$</td>
<td>9537, 9539, 9541, 9543, 9545, 9547, 9549, ...</td>
</tr>
</tbody>
</table>

From the table it is clear that $S \land LNF = STOF$.

Example for Theorem 5.5: Consider the following TCAS II specification given in Equation (5.2). Let the literal $e$ in the term $P_i = (a \land b \land c \land e \land f \land h \land \overline{n})$ in $S$ be replaced by the literal $d$ to get $LRF$. Let the same literal $e$ in the same term $P_i$ be negated to get $LNF$. The table below shows the test set for $SLRF$.
Table 5.15 Test set of $SLRF$ from Literal Negation Fault

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(S \land LNF \land d) \lor \overline{(S \land LNF \land d)}$</td>
<td>8512, 8514, 8518, 8520, 8522, 8526,...</td>
</tr>
<tr>
<td>$SLRF$</td>
<td>8512, 8514, 8518, 8520, 8522, 8526,...</td>
</tr>
</tbody>
</table>

From the table it is clear that $(S \land LNF \land x_t) \lor \overline{(S \land LNF \land x_t)} = SLRF$.

Example for Theorem 5.6: Consider the following TCAS II specification given in Equation (5.4). Let the literal $a$ in the term $P_i = (a \land c \land e \land f \land g)$ in $S$ be negated to get $LNF$. Let the specification $S$ be negated to get $ENF$. The table below shows the test set for $SLNF$ and $SENF$.

Table 5.16 Test set of $SENF$ from Literal Negation Fault

<table>
<thead>
<tr>
<th>Fault condition</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SLNF$</td>
<td>46, 47, 62, 63, 110, 111, 174, 175, 190, 191,...</td>
</tr>
<tr>
<td>$SENF$</td>
<td>0, 1, ..., 46, 47, ..., 62, 63, ..., 110, 111, ..., 174, 175, ..., 190, 191,...</td>
</tr>
</tbody>
</table>

From the table it is clear that if a test case can detect a Literal Negation Fault, it would also detect the Expression Negation Fault.

5.7 COMPARISON

A number of test set generation strategies are given in the literature. Almost all of them generated test cases from Unique True Points and Near False Points.

When Literal Negation Fault was analyzed, it was observed that the test cases that belonged to $S \land LNF$ gave the set of Unique True Points for the term in which a literal was negated. It was also observed that the test cases that belonged to
\( S \land LNF \) was equal to the test cases that belonged to the Near False Points. The following venn diagram compares the UTP and the NFP with \( S_{LNF} \).

![Venn Diagram](image)

Figure 5.2 Comparison of UTP and NFP with \( S_{LNF} \)

The following table gives the relationship of Literal Negation Fault with Unique True Points and Near False Points.

<table>
<thead>
<tr>
<th>Test set method proposed by Weyuker et al</th>
<th>Proposed test expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTP(_i)</td>
<td>( S \land LNF_{i,j} )</td>
</tr>
<tr>
<td>NFP(_{ij})</td>
<td>( \overline{S} \land LNF_{i,j} )</td>
</tr>
</tbody>
</table>

Here, \( i \) represents the \( i^{th} \) term in the specification and \( j \) represents the \( j^{th} \) literal in the \( i^{th} \) term. That is, for \( LNF_{i,j} \), the \( j^{th} \) literal in the \( i^{th} \) term is negated. UTP\(_i\) represents the Unique True Points of the \( i^{th} \) term. NFP\(_{ij}\) represents the Near False Points for the \( j^{th} \) literal in the \( i^{th} \) term.
The following example gives the relationship between Literal Negation Fault, Unique True Point and Near False Point. Consider the boolean specification given below.

\[ S = ((a \land b) \lor (c \land d)) \]  

(5.28)

Let the literal \( a \) in the term \( P_i \) in \( S \) be negated to get \( LNF \). The following table gives the Unique True Points and Near False Points along with the true points obtained by the expression \( S \land \overline{LNF} \) and \( \overline{S} \land \overline{LNF} \).

Table 5.18 Comparison of Literal Negation Fault with UTP and NFP

<table>
<thead>
<tr>
<th>Expression</th>
<th>True Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTP (_i)</td>
<td>12, 13, 14</td>
</tr>
<tr>
<td>( S \land \overline{LNF} )</td>
<td>12, 13, 14</td>
</tr>
<tr>
<td>NFP (_{1,1})</td>
<td>4, 5, 6</td>
</tr>
<tr>
<td>( \overline{S} \land \overline{LNF} )</td>
<td>4, 5, 6</td>
</tr>
</tbody>
</table>

It is clear from the above table that the set of Unique True Points and Near False Points can be generated from Literal Negation Fault.

5.8 SIGNIFICANCE OF THE RESULTS

In this chapter, the Null Fault is explained in detail with some examples. How Null fault occurs is explained and a method is proposed to detect the Null Fault. Further, certain exceptions that may occur in the hierarchy of fault classes are also explained using examples. The hierarchy to be followed in these situations is proposed.

Further, expressions are given to derive test sets for various faults from Literal Negation Fault. A comparison of the test cases of Literal Negation Fault and the Unique True Point and Near False Point is also provided. The need to go for
Literal Negation Fault to detect test cases for various faults is also presented. The test expressions to generate test sets for various faults from Literal Negation Fault are given in the following table.

**Table 5.19 Test expressions for various faults from Literal Negation Fault**

<table>
<thead>
<tr>
<th>Fault</th>
<th>Test Expression / Fault Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression Negation Fault</td>
<td>$SLNF \ (SLNF \Rightarrow \text{SENF})$</td>
</tr>
<tr>
<td>Term Negation Fault</td>
<td>$SLNF \ (SLNF \Rightarrow \text{STNF})$</td>
</tr>
<tr>
<td>Literal Negation Fault</td>
<td>$SLNF$</td>
</tr>
<tr>
<td>Term Omission Fault</td>
<td>$S \land \text{LNF}$</td>
</tr>
<tr>
<td>Literal Reference Fault</td>
<td>$(S \land \text{LNF} \land x_1) \lor (\overline{S} \land \text{LNF} \land x_2)$</td>
</tr>
<tr>
<td>Literal Omission Fault</td>
<td>$\overline{S} \land \text{LNF}$</td>
</tr>
<tr>
<td>Literal Insertion Fault</td>
<td>$S \land \text{LNF} \land \overline{x_1}$</td>
</tr>
<tr>
<td>Term Insertion Fault</td>
<td>$\overline{S} \land \text{LNF} \land \overline{x_1}$</td>
</tr>
</tbody>
</table>

Faults have some particular characteristics. Analyzing those characteristics can help in effectively deriving test sets to detect the specific fault. In the next chapter, the characteristics of faults are studied and a New Fault Condition is proposed which is computationally efficient and generates smaller test sets for some faults.