



## Research Note

RN/14/14

### **No Pot of Gold at the End of Program Spectrum Rainbow: Greatest Risk Evaluation Formula Does Not Exist**

November 3, 2014

*Shin Yoo*<sup>1</sup>, *Xiaoyuan Xie*<sup>2</sup>, *Fei-Ching Kuo*<sup>2</sup>, *Tsong Yueh Chen*<sup>2</sup>, *Mark Harman*<sup>1</sup>

*Affiliation:* University College London<sup>1</sup>, Swinburn University<sup>2</sup>

*E-Mail:* xxie@swin.edu.au, dkuo@swin.edu.au, tychen@swin.edu.au

shin.yoo@ucl.ac.uk, mark.harman@ucl.ac.uk

### **Abstract**

Spectrum Based Fault Localisation (SBFL) techniques rely on risk assessment formulæ to convert program execution spectrum into risk evaluation values, which are in turn used to rank program statements according to their relative suspiciousness with respect to the observed failure. Recent work proved equivalence and hierarchy between different formulæ, identifying a few groups of *maximal* formulæ, i.e., formulæ that do not dominate each other. The holy grail in the field has been to come up with the greatest formula, that is, the one that dominates all known formulæ. This paper proves that such a formula does not exist.

## 1 Introduction

Spectrum Based Fault Localisation (SBFL) is an automated debugging aid technique that assigns risk evaluation values to program statements. The values are calculated from the spectrum data, obtained during test execution. The most widely adopted form of the spectrum data is in the form of a tuple  $\sigma = (e_p, e_f, n_p, n_f)$ , collected per program statement:  $e_p$  and  $e_f$  represent the number of tests that execute the statement and pass and fail respectively, while  $n_p$  and  $n_f$  represent the number of tests that do not execute the statement and pass and fail respectively. A risk assessment formula  $R$  is a formula that converts the spectrum data into the risk evaluation value, based on which program statements are either ranked for humans [14] or assigned probabilities of being mutated for the purpose of automated program repair [16].

A large part of the literature focuses on designing and empirically evaluation different formulæ [1, 3, 9, 17]. More recently, Genetic Programming (GP) has been applied to automatically evolve risk evaluation formulæ [20].

Empirical evaluation of formulæ using known faults has been the de facto standard way of comparing performance of different formulæ. However, Recent work showed that theoretical comparison of formulæ is possible [11] and proved equivalence and hierarchy between formulæ groups [18].

A formula  $R_1$  is *better* than another formula  $R_2$  when it can be proved that  $R_1$  *always* ranks the faulty statement higher than or equal to  $R_2$ , *regardless* of the program, the test suite, and the fault. The proved hierarchy between formulæ can be summarised as a forest of directed trees [18]. There are multiple hierarchical trees, at the tops of which exist non-dominated, *maximal* groups. Formulæ from different maximal groups do not dominate each other. That is,  $R_1$  outperforms  $R_2$  for certain combinations of program, test suites, and faults, whereas  $R_2$  outperforms  $R_1$  for others.

Subsequently, the formulæ evolved by GP has been proved to be equivalent to the best known formulæ designed by human [19].

While the theoretical framework provided an alternative to empirical evaluation of different formulæ, there were limitations. First, the hierarchy was only proved within the set of 50 studied formulæ, and not the entire set of all possible formulæ. Second, it did not provide any insights into the existence of the *greatest* formula, i.e. one that would outperform all other formulæ.

This paper proves that the *greatest* formula, i.e. one that is better than all other formulæ, does not exist. The paper also proposes a novel visualisation technique that can present SBFL formulæ in an intuitive way.

## 2 Background

### 2.1 Spectrum-Based Fault Localisation

#### 2.1.1 Basic concept

Spectrum-Based Fault Localisation (SBFL) refers to a group of techniques that use program spectra to find the location of the fault in the given program that causes certain tests to fail. Program spectra can be best described as a summary of a set of program executions [5]. For the SBFL techniques, the most widely used type of program spectra is the combination of code coverage and the test results, on which this paper focuses too. Suppose SUT has  $n$  statements, and the test suite contains  $m$  test cases: the program spectrum for SBFL can be described as a matrix of  $n$  rows and 4 columns. Each row corresponds to individual statement of SUT, and contains the tuple  $(e_f, e_p, n_f, n_p)$ . Members  $e_f$  and  $e_p$  represent the number of times the corresponding program statement has been executed by tests, with fail and pass as a result respectively. Similarly,  $n_f$  and  $n_p$  represent the number of times the corresponding program statement has *not* been executed by tests, with fail and pass as a result respectively<sup>1</sup>. SBFL techniques subsequently use a risk

---

<sup>1</sup>The sum of  $e_f, e_p, n_f,$  and  $n_p$  should be  $m$ .

evaluation formula, which is a formula based on the four counters, to predict the relative risk of each statement containing the fault. Compared to the case in which the developer investigates the structural elements in the order from  $s_1$  to  $s_9$ , the ranking according to Tarantula produces 66.66% reduction in debugging effort (i.e. the developer will encounter  $s_7$  6 elements earlier).

Structural Elements	Test	Test	Test	Spectrum				Tarantula	Rank
	$t_1$	$t_2$	$t_3$	$e_f$	$e_p$	$n_f$	$n_p$		
$s_1$	•			0	1	2	0	0.00	9
$s_2$	•			0	1	2	0	0.00	9
$s_3$	•			0	1	2	0	0.00	9
$s_4$	•			0	1	2	0	0.00	9
$s_5$	•			0	1	2	0	0.00	9
$s_6$	•		•	1	1	1	0	0.33	4
$s_7$ (faulty)		•	•	2	0	0	1	1.00	1
$s_8$	•	•		1	1	1	0	0.33	4
$s_9$	•	•	•	2	1	0	0	0.50	2
Result	P	F	F						

Table 1: Motivating Example: the faulty statement  $s_7$  achieves the 1st place when ranked according to the Tarantula risk evaluation formula in Equation 1.

For example, Table 1 illustrates how the Tarantula metric [7], defined in Equation 1, can be applied to a small exemplar program spectrum. Suppose the structural element  $s_7$  contains the fault. The coverage relationship between structural elements and the given test suite  $T = \{t_1, t_2, t_3\}$  is given in the second column, with the corresponding test results. The Spectrum column contains the program spectrum data for  $T$ ; the column Tarantula contains the resulting risk evaluation metric values. Finally, the column Rank contains the ranking of structural elements according to the Tarantula metric values. The faulty statement,  $s_7$ , is assigned with the highest Tarantula metric value, and therefore ends up in the first place.

$$\text{Tarantula} = \frac{\frac{e_f}{e_f+n_f}}{\frac{e_p}{e_p+n_p} + \frac{e_f}{e_f+n_f}} \quad (1)$$

### 2.1.2 Risk evaluation formula

Based on Section 2.1.1, a SBFL risk evaluation formula is a function from program spectrum to suspiciousness score, such as Tarantula in Equation 1. More formally, it is defined as follow:

**Definition 2.1.** A risk evaluation formula  $R$  is a member of set  $\mathbb{R} = \{R | R : I \times I \times I \times I \rightarrow Real\}$  (where  $I$  denotes the set of non-negative integers and  $Real$  denotes the set of real numbers), which maps  $A_i = \langle e_f^i, e_p^i, n_f^i, n_p^i \rangle$  of each statement  $s_i$  to its risk value.

These formulæ are either designed by human [9, 11] or by Genetic Programming [20]. Table 2 contains several of the most widely studied formulæ. Interestingly, Jaccard [6] and Ochiai [12] were first studied in Botany and Zoology respectively but have been subsequently studied in the context of fault localisation [2, 11]. Tarantula was originally developed as a visualisation method [8, 9] but also increasingly considered as an SBFL risk evaluation formula independent from visualisation [7, 13]. AMPLE [4] and three different versions of Wong metric [17] have been introduced specifically for fault localisation. Lately, Genetic

Programming (GP) has been applied to SBFL: instead of being manually designed, risk evaluation formulæ were evolved by GP from given fault datasets [20].

Table 2: SBFL formulæ

	Name	Formula expression
$ER'_1$	Naish1	$\begin{cases} -1 & \text{if } e_f < F \\ n_p & \text{if } e_f = F \end{cases}$
	Naish2	$e_f - \frac{e_p}{e_p + n_p + 1}$
	GP13	$e_f \left(1 + \frac{1}{2e_p + e_f}\right)$
$ER_5$	Wong1	$e_f$
	Russel & Rao	$\frac{e_f}{e_f + n_f + e_p + n_p}$
	Binary	$\begin{cases} 0 & \text{if } e_f < F \\ 1 & \text{if } e_f = F \end{cases}$
	GP02	$2(e_f + \sqrt{n_p}) + \sqrt{e_p}$
	GP03	$\sqrt{ e_f^2 - \sqrt{e_p} }$
	GP19	$e_f \sqrt{ e_p - e_f + n_f - n_p }$

Naish1 and Naish2 metrics are recent additions to SBFL techniques that showed an interesting research direction: these metrics are designed with accompanying proof that shows they produce optimal ranking, as long as the fault is located in a specific program structure (two consecutive If-Then-Else blocks, called ITE2) [11]. It was the first attempt at theoretical analysis of SBFL formulæ. Subsequently, Xie et al. [18] proved the hierarchy between formulæ, showing that there exist multiple hierarchy trees. Some formulæ were proven to be equivalent to others (i.e. belong to the same node in the hierarchy tree) with respect to the ranking of the faulty statement. The formulæ at the top are called *maximal* because they dominated others (i.e. always produce higher ranking for the faulty statement than the other) in the tree. However, no globally maximal formula is known; all the known maximals did not dominate each other. Some of the GP evolved formulæ [20] were proven to be maximal, but not globally maximal.

## 2.2 Theoretical framework

With the development of more and more risk evaluation formulæ, people began to investigate their performance. The most commonly adopted effectiveness measurement is referred to as Expense metric, which is the percentage of code that needs to be examined before the faulty statement is identified [20]. A lower Expense of formula  $R$  indicates a better performance. Recently a theoretical framework [18] has been developed to compare the performances of different formulæ against any combinations of programs, faulty statements, and consistent tie-breaking schemes<sup>2</sup>. The consistent tie-breaking scheme, and the relationships between different formulæ, are defined as follows:

**Definition 2.2** (Consistent tie-breaking scheme). Given any two sets of statements  $S_1$  and  $S_2$ , which contain elements having the same risk values. A tie-breaking scheme returns the ordered statement lists  $O_1$  and

<sup>2</sup>In practice, a tie-breaking scheme may be required to determine the order of the statements with same risk values.

$O_2$  for  $S_1$  and  $S_2$ , respectively. The tie-breaking scheme is said to be consistent, if all elements common to  $S_1$  and  $S_2$  have the same relative order in both  $O_1$  and  $O_2$ .

Let  $R_1$  and  $R_2$  be two risk evaluation formulæ in  $\mathbb{R}$ , and  $E_1$  and  $E_2$  denote the Expenses with respect to the same faulty statement for  $R_1$  and  $R_2$ , respectively. We define two types of relations between  $R_1$  and  $R_2$  as follows.

**Definition 2.3** (Better).  $R_1$  is said to be *better* than  $R_2$  (denoted as  $R_1 \rightarrow R_2$ ) if, for any program, faulty statement  $s_f$ , test suite, and consistent tie-breaking scheme,  $E_1$  is less than or equal to  $E_2$ .

**Definition 2.4** (Equivalent).  $R_1$  and  $R_2$  are said to be *equivalent* (denoted as  $R_1 \leftrightarrow R_2$ ), if, for any program, faulty statement  $s_f$ , test suite and consistent tie-breaking scheme,  $E_1$  is equal to  $E_2$ .

It follows from the definition that  $R_1 \rightarrow R_2$  means  $R_2$  is not more effective than  $R_1$ . As a reminder, if both  $R_1 \rightarrow R_2$  and  $R_2 \rightarrow R_1$  hold, then it follows that  $R_1 \leftrightarrow R_2$ ; if  $R_1 \rightarrow R_2$  holds but  $R_2 \rightarrow R_1$  does not hold,  $R_1 \rightarrow R_2$  is said to be a strictly “*better*” relation.

In the theoretical framework, there are several assumptions, which are listed as follows:

1. A test oracle exists, that is, for any test case, the testing result of either *fail* or *pass*, can be decided.
2. We exclude two types of faults that SBFL is not designed for. The first type includes omission faults. We assume that for all observed failures, the execution of a faulty statement  $s_f$  is the cause. In other words, a statement cannot cause a failure by not being executed. A more generalised statement would be that SBFL techniques cannot localise omission faults (i.e. statements forgotten by the programmer, such as a missing null check). The second type includes the non-deterministic faults. SBFL techniques expect the same test input to produce the same program spectrum with every execution.
3. The fault is executed by the test suite. Being a type of dynamic analysis, SBFL techniques cannot localise faults in statements that are not covered by the test suite.
4. For each fault that needs to be localised, the test suite contains at least one passing test case and one failing test case.

As a reminder, our analysis only focuses on statements that are covered by the given test suite, since only these statements are possible to be the faulty statement that triggers the observed failure. And thus the statements that is never executed by any test case in the given test suite should be ignored or assigned with the lowest risk values. Moreover, our analysis only consider programs with single fault. For readers who are interested in the justifications, validity and impacts of the above assumptions, please refer to the previous work [18].

In order to compare two risk evaluation formulæ in  $\mathbb{R}$  under the above definitions of relations, the previous work [18] have provided a theoretical framework, which divides all statements into three mutually exclusive subsets, as follows.

**Definition 2.5.** Given a program with  $n$  statements  $PG = \langle s_1, s_2, \dots, s_n \rangle$ , a test suite of  $m$  test cases  $TS = \{t_1, t_2, \dots, t_m\}$ , and a risk evaluation formula  $R$ , which assigns a risk value to each program statement. For each statement  $s_i$ , a spectrum vector  $\sigma(s_i) = \langle e_f^i, e_p^i, n_f^i, n_p^i \rangle$  can be constructed from  $TS$ , and  $R(e_f^i, e_p^i, n_f^i, n_p^i)$  is a risk evaluation formula that assigns a risk value to statement  $s_i$ . For any faulty statement  $s_f$ , it is possible to define the following three sets of statements:

$$\begin{aligned} S_B^R &= \{s_i \in S \mid R(e_f^i, e_p^i, n_f^i, n_p^i) > R(e_f^f, e_p^f, n_f^f, n_p^f), 1 \leq i \leq n\} \\ S_F^R &= \{s_i \in S \mid R(e_f^i, e_p^i, n_f^i, n_p^i) = R(e_f^f, e_p^f, n_f^f, n_p^f), 1 \leq i \leq n\} \\ S_A^R &= \{s_i \in S \mid R(e_f^i, e_p^i, n_f^i, n_p^i) < R(e_f^f, e_p^f, n_f^f, n_p^f), 1 \leq i \leq n\} \end{aligned}$$

That is, statements in  $S_B^R$  have higher risk values than  $s_f$ , and thus are all ranked above any statements in  $S_F^R$ ; statements in  $S_F^R$  have the same equal risk value as that of  $s_f$  and, thus, are all ranked in the middle of the ranking list, together with  $s_f$  (tie-breaking scheme is needed to further distinguish them); and statements in  $S_A^R$  have lower risk values than  $s_f$  and, thus, are all ranked below any statements in  $S_F^R$ .

In the current framework, two results have been developed for establishing the relationship between two risk evaluation formulæ. They are as follows:

**Theorem 2.6.** Given any two risk evaluation formulæ  $R_1$  and  $R_2$  from  $\mathbb{R}$ , if, for any program, faulty statement  $s_f$ , and test suite, it holds that  $S_B^{R_1} \subseteq S_B^{R_2} \wedge S_A^{R_2} \subseteq S_A^{R_1}$ , then  $R_1 \rightarrow R_2$ .

**Theorem 2.7.** Let  $R_1$  and  $R_2$  be two risk evaluation formulæ from  $\mathbb{R}$ . If, for any program, faulty statement  $s_f$ , and test suite, it holds that  $S_B^{R_1} = S_B^{R_2} \wedge S_F^{R_1} = S_F^{R_2} \wedge S_A^{R_1} = S_A^{R_2}$ , then  $R_1 \leftrightarrow R_2$ .

With the theoretical framework, Xie et al. have investigated 30 formulæ, among which six equivalent groups of formulæ (namely,  $ER_1$  to  $ER_6$ ) have been identified and two of them are maximal groups of formulæ to the investigated formulæ [18]. The detailed and complete proofs that formulæ within each group share the same set subdivision can be found in the previous work [18]. The definition of limited maximality, i.e. maximality with respect to  $\mathbb{S}$ , is as follows:

**Definition 2.8. Limited Maximality.** A risk evaluation formula  $R_1$  from a subset of formulæ,  $\mathbb{S} \subset \mathbb{R}$ , is said to be a maximal formula of  $\mathbb{S}$  if for any element  $R_2 \in \mathbb{S}$ ,  $R_2 \rightarrow R_1$  implies  $R_2 \leftrightarrow R_1$ .

Table 3: Definition of investigated formulas

Name		Formula as $R(e_f^i, e_p^i, n_f^i, n_p^i)$	$\overline{R}(e_f^i, n_p^i)$ with $(P, F)$
$ER'_1$	Naish1	$\begin{cases} -1 & \text{if } e_f < F \\ n_p & \text{if } e_f = F \end{cases}$	$\begin{cases} -1 & \text{if } e_f < F \\ n_p & \text{if } e_f = F \end{cases}$
	Naish2	$e_f - \frac{e_p}{e_p + n_p + 1}$	$e_f - \frac{P - n_p}{P + 1}$
	GP13	$e_f(1 + \frac{1}{2e_p + e_f})$	$e_f(1 + \frac{1}{2(P - n_p) + e_f})$
$ER_5$	Wong1	$e_f$	$e_f$
	Russel & Rao	$\frac{e_f}{e_f + n_f + e_p + n_p}$	$\frac{e_f}{F + P}$
	Binary	$\begin{cases} 0 & \text{if } e_f < F \\ 1 & \text{if } e_f = F \end{cases}$	$\begin{cases} 0 & \text{if } e_f < F \\ 1 & \text{if } e_f = F \end{cases}$
GP2	$2(e_f + \sqrt{n_p}) + \sqrt{e_p}$	$2(e_f + \sqrt{n_p}) + \sqrt{P - n_p}$	
GP3	$\sqrt{ e_f^2 - \sqrt{e_p} }$	$\sqrt{ e_f^2 - \sqrt{P - n_p} }$	
GP19	$e_f \sqrt{ e_p - e_f + n_f - n_p }$	$e_f \sqrt{ F + P - 2e_f - 2n_p }$	

### 3 Maximal and Greatest Formulæ

The existing definition of a maximal formula in Definition 2.8 only concerned a subset of formulæ,  $\mathbb{S}$ , out of all possible formulæ,  $\mathbb{R}$ . The subset  $\mathbb{S}$  contained only 50 formulæ, 30 manually designed ones and 20 GP-evolved ones. The five identified maximal groups are only with respect to these 50 formulæ. Now, let us generalise our analysis by replacing  $\mathbb{S}$  with  $\mathbb{R}$ . This will, in turn, lead to the investigation of the “greatest” formula.

#### 3.1 Preliminaries

##### 3.1.1 Spectral Coordinate

Let us first present some definitions and lemmas. Given a test suite  $TS$ , let  $T$  denote its size,  $F$  denote the number of *failed* test cases and  $P$  denote the number of *passed* test cases. From the definitions and the earlier assumptions, it follows that  $1 \leq F < T$ ,  $1 \leq P < T$ , and  $P + F = T$ , as well as the following lemmas:

**Lemma 3.1.** For any  $\sigma(s_i) = \langle e_f^i, e_p^i, n_f^i, n_p^i \rangle$ , it holds that  $e_f^i + e_p^i > 0 \wedge e_f^i + n_f^i = F \wedge e_p^i + n_p^i = P \wedge e_f^i \leq F \wedge e_p^i \leq P$ .

**Lemma 3.2.** For any faulty statement  $s_f$  with  $\sigma(s_f) = \langle e_f^f, e_p^f, n_f^f, n_p^f \rangle$ , if  $s_f$  is the only faulty statement in the program, it follows that  $e_f^f = F \wedge n_f^f = 0$ .

For a given pair of program and test suite, the values of  $F$  and  $P$  are constants. Thus for each statement  $s_i$ , it follows that  $\sigma(s_i) = \langle e_f^i, P - n_p^i, F - e_f^i, n_p^i \rangle$  after Lemma 3.1, which can be denoted as  $\bar{\sigma}(s_i) = \langle e_f^i, e_p^i \rangle$ . That is, program spectrum contains two independent parameters in a specific context (i.e. a pair of a program and a test suite), and not four.

Consequently, it is possible to formulate  $\bar{\mathbb{R}} = \{\bar{R}|\bar{R} : I_f \times I_p \rightarrow Real\}$ , where  $I_f$  denotes the set of integers within  $[0, F]$  and  $I_p$  denotes the set of integers within  $[0, P]$ , such that  $\bar{R}(e_f^i, n_p^i) = R(e_f^i, e_p^i, n_f^i, n_p^i)$ . In the subsequent discussion, when two formulæ from  $\mathbb{R}$  are compared, it is assumed that they are being applied to the same program and test suite. Thus, in the context of such comparisons, symbols  $R$  and  $\bar{R}$  can and will be used interchangeably, as are symbols  $\mathbb{R}$  and  $\bar{\mathbb{R}}$ .

Given any values of  $P$  and  $F$ , the input domain of any formula  $\bar{R}$  is shown as the grid in Figure 1a, where both  $e_f$  and  $e_p$  are non-negative integers and  $0 \leq e_f^i \leq F$  and  $0 \leq e_p^i \leq P$ . Given a pair of test suite and program, each point  $(e_f, e_p)$  on this grid is associated with a group of statements that have the corresponding  $e_f$  and  $e_p$  values. Note that the number of statements that associated with each point  $(e_f, e_p)$  is independent of the formula, but solely decided by the pair of program and test suite.

A formula  $\bar{R}$  maps each point  $(e_f, e_p)$  to a real number that is the risk value of all statements associated with this point, as shown in Figure 1b. Any assignment of risk values is independent of the number of statements associated with each point  $(e_f, e_p)$ , but solely decided by the definition of  $\bar{R}$ .

##### 3.1.2 Analysis of SBFL Space

Lemma 3.2 allows us to limit the region of the input domain  $\bar{A}$  in which the faulty statement can be.

**Definition 3.3** (Faulty Border). Let us call the sequential points  $\langle (F, 0), (F, 1), \dots, (F, e_p), \dots, (F, P) \rangle$  ( $0 \leq e_p \leq P$ ) the *Faulty Border*, which is denoted as  $E$ . Figure 1b illustrates a potential  $E$ .

Immediately from the above definition, for any given formula  $R$ , it follows that the risk values of all points on  $E$  are solely decided by their values of  $e_p$ . And immediately after Lemma 3.2, the faulty statement  $s_f$  is associated with the point  $(F, e_p^f)$  of  $E$ , where  $0 \leq e_p^f \leq P$ , as stated in the following lemma.

**Lemma 3.4** (Location of faulty statement  $s_f$ ). The faulty statement  $s_f$  must be associated with a point  $(F, e_p^f)$  on  $E$ . And  $e_p^f$  can be any value between  $[0, P]$ .

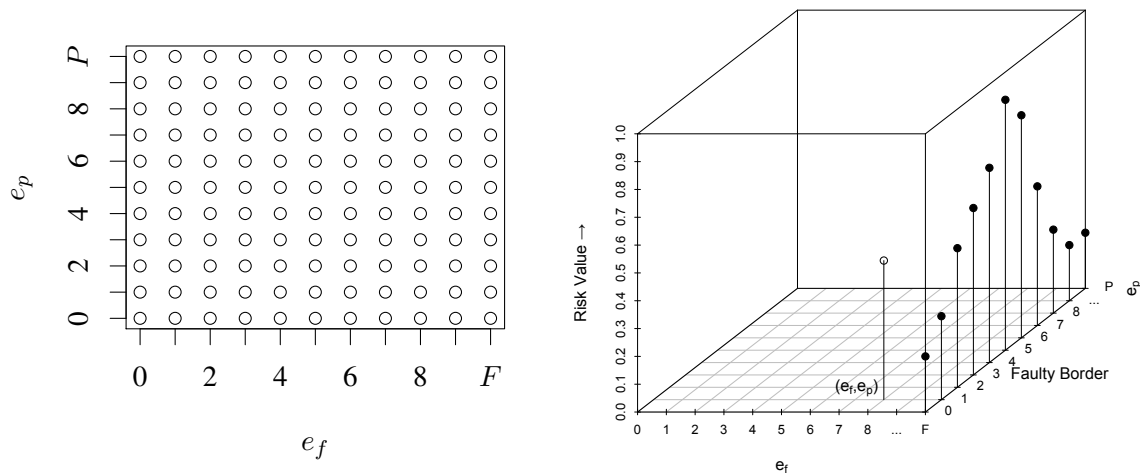
(a) Spectral Coordinate  $\bar{\sigma}$  for SBFL formulae(b) Mapping from  $\bar{\sigma}$  to risk values by formula  $\bar{R}$ 

Figure 1: Visualising the SBFL Space

Lemma 3.4 reflects a phenomenon in software testing called ‘‘Coincidental Correctness Test’’ (CCT). Ideally, the faulty statement  $s_f$  will produce  $e_p = 0$ , as executing  $s_f$  should result in a failure. CCTs are the tests that execute  $s_f$  but still pass. The number of CCT is equal to  $e_p^f$ , i.e. the value of  $e_p$  for  $s_f$ . There can be an arbitrary number of CCTs in a given test suite, and so is the value of  $e_p^f$ .

As a reminder, points  $(F, e_p^i)$  other than the one associated with  $s_f$  on  $E$  are associated with correct statements, where  $n_f^i = F \wedge 0 \leq e_p^i \leq P \wedge e_p^i \neq e_p^f$ . Depending on the adopted formula, the risk values of such points can be either greater than, equal to, or smaller than that of point  $(F, e_p^f)$ , i.e. the point associated with  $s_f$ .

**Lemma 3.5.** For a given program and a test suite, the point of  $E$ , with which  $s_f$  is associated, may also be associated with other correct statements  $s_i$  having  $(F, e_p^i) = (F, e_p^f)$ . These statements share the same risk values as that of  $s_f$ , regardless of the selection of the formula.

Lemma 3.5 reflects another common phenomenon in software testing, that is, correct statements  $s_i$  may still have  $e_f^i = F$ , and their  $e_p^i$  could be either greater than, equal to or smaller than  $e_p^f$  of the faulty statement  $s_f$  and so are their risk values. An example of the faulty border can be found in Figure 2.

### 3.2 Maximality in $\mathbb{R}$

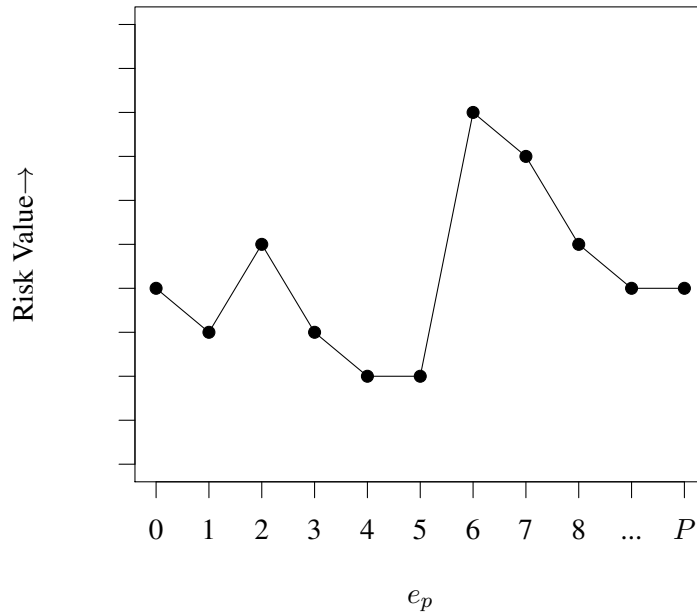
First, let us present the definition of the maximality with respect to  $\mathbb{R}$ .

**Definition 3.6. Maximality.** A risk evaluation formula  $R$  is said to be a maximal formula in  $\mathbb{R}$  if, for any formula  $R' \in \mathbb{R}$  such that  $R' \neq R \wedge R' \rightarrow R$ , it also holds that  $R' \leftrightarrow R$ . Let  $\mathbb{M}$  be the set of formulas that are maximal with respect to  $\mathbb{R}$ .

**Definition 3.7. Ranking.** Given a formula  $R$ , use  $o_p^{i,j} = \langle n_p^i, n_p^j, op \rangle$  to denote the relation between the risk scores of two distinct points  $(F, n_p^i)$  and  $(F, n_p^j)$  on the faulty border. Given that  $n_p^i < n_p^j$ ,  $op$  can be either ‘‘>’’ (i.e.,  $R(F, n_p^i) > R(F, n_p^j)$ ), ‘‘<’’ (i.e.,  $R(F, n_p^i) < R(F, n_p^j)$ ), or ‘‘=’’ (i.e.,  $R(F, n_p^i) = R(F, n_p^j)$ ).

Let  $P_R$  denote the set of  $o_p^{i,j}$  based on Definition 3.7.  $P_R$  effectively captures the ranking between the statements that belong to  $E$ , by collecting the relations between risk scores of each pair of distinct points



Figure 2: Example of faulty border of  $\bar{R}$ 

on the faulty border. Let  $U_R$  denote the set of points outside the faulty border which have risk scores higher than or equal to those of some points  $(F, e_p^i)$  on the faulty border, for formula  $R$ .

With all the above preliminary, let us now turn to the analysis of the maximality for all formulæ in  $\mathbb{R}$ .

**Lemma 3.8.** For any formula  $R \in \mathbb{R}$ , if  $U_R \neq \emptyset$ , then  $R \notin \mathbb{M}$ .

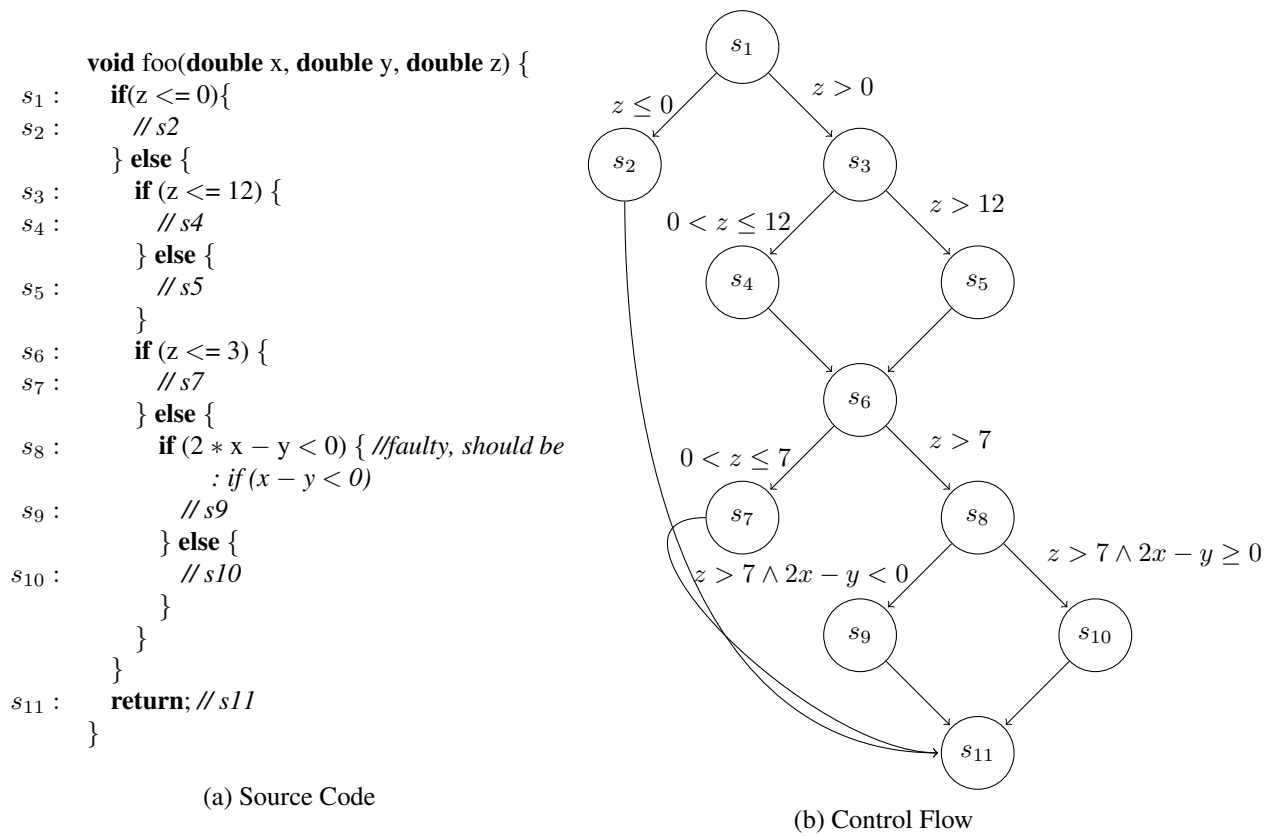
*Proof.* The proof shows that, if  $U_R \neq \emptyset$ , then there exists  $R' \in \mathbb{R}$  such that  $R' \rightarrow R$  but  $R \not\rightarrow R'$ . First, let us construct  $R' \in \mathbb{R}$  such that  $R' \rightarrow R$ . Assume that  $U_R$  is non-empty. Let  $R'$  be defined as follow:

$$R' = \begin{cases} R & \text{if } e_f = F \\ R - (C_1 - C_2 + 1) & \text{otherwise} \end{cases}$$

where  $C_1$  is the highest risk value of  $R$  for all points outside  $E$ , while  $C_2$  is the lowest risk value of  $R$  for all points on  $E$ . By the definition of  $R'$ , any point outside  $E$  has risk value lower than those of the points in  $E$ , which means all statements associated with points outside  $E$  have risk values lower than that of  $s_f$ .

Let  $U_{R'}$  denote the sets of points outside the faulty border which have risk values higher than or equal to those of some points  $(F, e_p^i)$  on the faulty border, for formula  $R'$ . By definition,  $R'$  assigns identical risk values to points on the faulty border as  $R$ , while ensuring that  $U_{R'} = \emptyset$ .

- Consider the statements associated with  $E$ : these statements will be assigned to the same set division by both  $R$  and  $R'$ , for any pair of program and test suite.
- Consider the statements associated with points outside  $E$ . For formula  $R'$ , since these points (including those in  $U_R$ ) always have risk values lower than that of  $s_f$  on  $E$ , the corresponding statements belong to  $S_A^{R'}$ . However, for formula  $R$ , since  $U_R \neq \emptyset$ , some statements corresponding to points outside  $E$  belong to either  $S_B^R$ ,  $S_F^R$ , and  $S_A^R$ .

Figure 3: Sample program: the faulty statement  $s_f$  is  $s_8$ .

Summarizing the above two cases, we have  $S_B^{R'} \subseteq S_B^R$  and  $S_A^R \subseteq S_A^{R'}$ . Following Theorem 2.6,  $R' \rightarrow R$ .

Let us now turn to show that  $R \not\rightarrow R'$ , by illustrating that it is possible for  $R'$  to produce a smaller Expense value than  $R$ . Since  $U_R \neq \emptyset$ , there exists  $L$ , a set of points on  $E$  whose risk values evaluated by  $R$  are not higher than any point in  $U_R$ . To show that  $R'$  can produce a smaller Expense value than  $R$ , it is sufficient to show that  $\bar{\sigma}(s_f) \in L$  while  $U_R \neq \emptyset$ . However, both  $L$  and  $U_R$  are specific to the choice of  $R$ . In order not to lose generality, therefore, let us show that it is possible to construct a program and a test suite such that  $\bar{\sigma}(s_f)$  can be placed anywhere on  $E$ , and another statement  $\bar{\sigma}(s_i)$  can be placed anywhere in  $I_f \times I_p - E$ , independently from each other<sup>3</sup>. Figure 3 illustrates such a program: the feasibility of the construction of the test suite is described in Example 1 in Appendix.

With such a program and a test suite, any statement associated with points outside  $U_R$  always have the same relative ranking to  $s_f$  in  $R$  and  $R'$ . For all statements associated with  $U_R$ , formula  $R'$  will rank them below  $s_f$ . However, with  $R$ :

- statements that are associated with  $U_R$  and have risk values higher than that of  $s_f$ , are always ranked before  $s_f$  by  $R$ .
- statements that are associated with  $U_R$  and have risk values equal to that of  $s_f$ , will be tied together with  $s_f$  by  $R$ . However, it is possible to have a consistent tie-breaking scheme which ranks parts or even all of these statements before  $s_f$ .

It is always possible to have statements associated with  $U_R$  ranked before  $s_f$ . Consequently, the Expense of  $R'$  is smaller than that of  $R$ . Therefore,  $R \rightarrow R'$  does hold.

<sup>3</sup>Given a specific  $R$  such that  $U_R \neq \emptyset$ , this allows us to place  $\bar{\sigma}(s_f) \in L$  and  $\bar{\sigma}(s_i) \in U_R$ .

In conclusion, if  $R$  assigns point  $(e_f^j, e_p^j)$  outside  $E$  with risk value higher than, or equal to, that of at least one point  $(F, e_p^i)$  on  $E$ , there always exists another formula  $R'$  for which  $R' \rightarrow R$  holds but  $R \rightarrow R'$  does not hold. Therefore, following Definition 3.6,  $R$  cannot be a maximal formula.  $\square$

Given two distinct risk evaluation formulæ,  $R_1$  and  $R_2$ , let  $P_{R_1}$  and  $P_{R_2}$  denote the set of  $o_p^{i,j}$  for all pairs of distinct points  $(F, e_p^i)$  and  $(F, e_p^j)$  on the faulty border (where  $e_p^i < e_p^j$ ), for  $R_1$  and  $R_2$ , respectively. Let  $U_{R_1}$  and  $U_{R_2}$  denote the sets of points outside the faulty border which have risk values higher than, or equal to, those of some points  $(F, e_p^i)$  on the faulty border, for formula  $R_1$  and  $R_2$ , respectively.

**Lemma 3.9.** If  $U_{R_1} = U_{R_2} = \emptyset$  and  $P_{R_1} = P_{R_2}$ , it follows that  $R_1 \leftrightarrow R_2$ .

*Proof.* Consider the following two cases.

- For statements associated with the faulty border  $E$ , since  $P_{R_1} = P_{R_2}$ , then for each pair of these statements, the relation between their risk values is always the same in  $R_1$  and  $R_2$ . As a consequence, these statements have the same relative order with respect to  $s_f$  (which is associated with one point on the faulty border) between  $R_1$  and  $R_2$ , and hence belong to the same set-division for  $R_1$  and  $R_2$  with any pair of program and test suite.
- For statements associated with points outside  $E$ , since both  $U_{R_1}$  and  $U_{R_2}$  are empty, these statements always have risk values lower than that of the faulty statement  $s_f$  (which is associated with one point on the faulty border), therefore these statements belong to both  $S_A^{R_1}$  and  $S_A^{R_2}$ .

In summary, we have  $S_B^{R_1} = S_B^{R_2}$ ,  $S_F^{R_1} = S_F^{R_2}$  and  $S_A^{R_1} = S_A^{R_2}$ . Following Theorem 2.7,  $R_1 \leftrightarrow R_2$ .  $\square$

**Lemma 3.10.** If  $U_{R_1} = U_{R_2} = \emptyset$  but  $P_{R_1} \neq P_{R_2}$ , we have  $R_1 \nrightarrow R_2$  and  $R_2 \nrightarrow R_1$ .

*Proof.* Since  $P_{R_1} \neq P_{R_2}$ , there must exist at least one pair of points on the faulty border  $((F, e_p^i), (F, e_p^j))$  (where  $e_p^i < e_p^j$ ), such that  $\langle e_p^i, e_p^j, op_1 \rangle \in P_{R_1} \wedge \langle e_p^i, e_p^j, op_2 \rangle \in P_{R_2} \wedge op_1 \neq op_2$ . It is sufficient to consider the following two cases because other cases can be transformed to these two cases by swapping  $R_1$  and  $R_2$ :

- Consider the case that  $R_1(F, e_p^i) < R_1(F, e_p^j)$  and  $R_2(F, e_p^i) > R_2(F, e_p^j)$ .

With the program shown in Figure 3, it is possible to construct a test suite, such that  $e_f^4, e_f^5, e_f^9$  and  $e_f^{10}$  are smaller than  $F$ . (As a reminder, it always holds that  $e_f^2 = e_f^7 = 0$ ). While for  $s_1, s_3, s_6, s_8$  ( $s_f$ ) and  $s_{11}$ , whose  $e_f$  values are all equal to  $F$ , we have  $e_p^f = e_p^i < e_p^1 = e_p^3 = e_p^6 = e_p^{11} = e_p^4$ . Then, for  $R_1$ , we have  $s_1, s_3, s_6$  and  $s_{11}$  ranked before  $s_f$  and other statements ranked after  $s_f$ . However, for  $R_2$ , we have  $s_f$  ranked at the top of the whole list. Therefore, the Expense of  $R_2$  is lower than that of  $R_1$ .

On the other hand, it is also possible to construct another test suite, such that  $e_f^4$  and  $e_f^5$  are both smaller than  $F$ , but  $e_f^9$  is equal to  $F$ . (Correspondingly,  $e_f^{10} = 0$ ). For  $s_1, s_3, s_6, s_8$  ( $s_f$ ),  $s_9$  and  $s_{11}$ , whose  $e_f$  values are all equal to  $F$ , we have  $e_p^9 = e_p^i < e_p^1 = e_p^3 = e_p^6 = e_p^f = e_p^{11} = e_p^4$ . Then, for  $R_1$ ,  $s_1, s_3, s_6, s_f$  and  $s_{11}$  are tied together at the top of the whole list, before  $s_9$ . However, for  $R_2$ ,  $s_9$  is ranked at the top, immediately followed by  $s_1, s_3, s_6, s_f$  and  $s_{11}$  that are tied together. Therefore, with a consistent tie-breaking scheme, the Expense of  $R_1$  is lower than that of  $R_2$ .

In summary, for the case that  $R_1(F, e_p^i) < R_1(F, e_p^j)$  while  $R_2(F, e_p^i) > R_2(F, e_p^j)$ , it is always possible to find examples to demonstrate  $R_1 \nrightarrow R_2$  and  $R_2 \nrightarrow R_1$

<sup>4</sup>For the feasibility of this test suite, please refer to **Test Suite A** in Example 5 of the Appendix.

<sup>5</sup>For the feasibility of this test suite, please refer to **Test Suite B** in Example 5 of the Appendix.

- Consider the case that  $R_1(F, e_p^i) < R_1(F, e_p^j)$  and  $R_2(F, e_p^i) = R_2(F, e_p^j)$ .

With the program shown in Figure 3, it is possible to construct a test suite, such that  $e_f^4 = F$  (correspondingly,  $e_f^5 = 0$ ), while  $e_f^9$  and  $e_f^{10}$  are smaller than  $F$ . Then, for  $s_1, s_3, s_4, s_6, s_8 (s_f)$ , and  $s_{11}$ , whose  $e_f$  values are all equal to  $F$ , it follows that  $e_p^4 = e_p^i < e_p^1 = e_p^3 = e_p^6 = e_p^f = e_p^{11} = e_p^{j6}$ . Then, for  $R_1$ ,  $s_1, s_3, s_6, s_f$ , and  $s_{11}$  are tied together at the top of the ranking, before  $s_4$ . However, for  $R_2$ ,  $s_1, s_3, s_4, s_6, s_f$ , and  $s_{11}$  are tied together at the top of the entire ranking. Since the number of tied statements are different, the Expense now depends on the tie-breaking scheme, without any guarantee of clear dominance of one formula. For example, if the original order of the statements is used as the tie-breaker,  $R_1$  yields a lower Expense value than  $R_2$ ; if the reverse of the original order is adopted, the opposite would follow.

On the other hand, it is also possible to construct another test suite, such that  $e_f^4, e_f^5, e_f^9$  and  $e_f^{10}$  are smaller than  $F$ . Then, for  $s_1, s_3, s_6, s_8 (s_f)$  and  $s_{11}$  whose  $e_f$  values are all equal to  $F$ , we have  $e_p^f = e_p^i < e_p^1 = e_p^3 = e_p^6 = e_p^{11} = e_p^j$ . (For the feasibility of this test suite, please refer to **Test Suite A** in Example 5 of the Appendix.) Then, for  $R_1$ , we have  $s_1, s_3, s_6$  and  $s_{11}$  tied together at the top of the whole list before  $s_f$ , and other statements ranked after  $s_f$ . However, for  $R_2$ , we have  $s_1, s_3, s_6, s_f$  and  $s_{11}$  tied together at the top of the whole list. Since the number of tied statements are different, the Expense now depends on the tie-breaking scheme, without any guarantee of clear dominance of one formula. For example, if the original order of the statements is used as the tie-breaker,  $R_2$  yields a lower Expense value than  $R_1$ ; if the reverse of the original order is adopted, the opposite would follow.

In summary, for the case that  $R_1(F, e_p^i) < R_1(F, e_p^j)$  while  $R_2(F, e_p^i) = R_2(F, e_p^j)$ , it is possible to demonstrate that  $R_1 \not\rightarrow R_2$  and  $R_2 \not\rightarrow R_1$ .

In conclusion, for any two formulæ whose  $U_{R_1}$  and  $U_{R_2}$  are both  $\emptyset$ , but  $P_{R_1} \neq P_{R_2}$ , it follows that  $R_1 \not\rightarrow R_2$  and  $R_2 \not\rightarrow R_1$ .  $\square$

With above preliminaries, let us now turn to the complete analysis of the maximal and greatest formulæ of  $\mathbb{R}$ .

**Proposition 3.11.** A formula  $R$  is a maximal formula of in  $\mathbb{R}$  if and only if  $U_R$  is empty.

*Proof.* First, it holds that if  $R$  is a maximal formula, then  $U_R = \emptyset$ . This follows immediately after Lemma 3.8.

Second, let us turn to proving that if  $U_R = \emptyset$ ,  $R$  is a maximal formula. Assume that  $U_R = \emptyset$ . Then, for any distinct formula  $R'$ , let  $P_{R'}$  denote the set of  $o_p^{i,j}$  for all pairs of distinct points  $(F, n_p^i)$  and  $(F, n_p^j)$  on the faulty border (where  $n_p^i < n_p^j$ ) and  $U_{R'}$  denote the sets of points outside the faulty border which have risk values higher than or equal to those of some points  $(F, n_p^i)$  on the faulty border, for formula  $R'$ . There are following cases.

- Consider the case that  $U_{R'} \neq \emptyset$ . As illustrated in the proof of Lemma 3.8, it is always possible to construct another formula  $R''$ , such that  $U_{R''} = \emptyset$ ,  $R'' \rightarrow R'$  and  $R' \not\rightarrow R''$ . If  $P_{R''} = P_R$ , after Lemma 3.9,  $R'' \leftrightarrow R$ , and, consequently,  $R' \not\rightarrow R$ . Otherwise, if  $P_{R''} \neq P_R$ , after Lemma 3.10,  $R \not\rightarrow R''$  and  $R'' \not\rightarrow R$ . As a consequence,  $R' \not\rightarrow R$ .
- Consider the case that  $U_{R'} = \emptyset$ . Similar to the above analysis, if  $P_{R'} = P_R$ , after Lemma 3.9,  $R' \leftrightarrow R$ . Otherwise, if  $P_{R'} \neq P_R$ , after Lemma 3.10,  $R \not\rightarrow R'$  and  $R' \not\rightarrow R$ .

In summary, if  $U_R = \emptyset$ , then for any formula  $R'$ , we have either  $R' \leftrightarrow R$  or  $R' \not\rightarrow R$ . After Definition 3.6,  $R$  is a maximal formula.  $\square$

<sup>6</sup>For the feasibility of this test suite, please refer to **Test Suite C** in Example 5 of the Appendix.

With Proposition 3.11, which is a necessary and sufficient condition for a maximal formula of  $\mathbb{R}$ , there is a simple method to convert any given non-maximal formula into a maximal formula, which is effectively described in the proof of Lemma 3.8. Given any formula  $R$  which has a non-empty  $U_R$  (i.e.  $R$  is non-maximal according to Proposition 3.11), we can always convert  $R$  into a maximal formula  $R'$ , where  $R'$  assigns identical risk values to points on the faulty border as  $R$ , but assigns, to all points in  $U_R$ , a constant  $C$  whose value is smaller than the risk value of any point on the faulty border.

Now, let us re-visit the five maximal formulæ of the 50 investigated formulæ. After Proposition 3.11, it is possible to show that two of them ( $ER'_1$  and  $ER_5$ ) are maximal but not greatest formulæ, while three of them (GP2, GP3 and GP19) are not maximal with respect to  $\mathbb{R}$ , as follows:

**Corollary 3.12.**  $ER'_1$  and  $ER_5$  are maximal formulæ of  $\mathbb{R}$ .

*Proof.* For any formula  $R$  in  $ER'_1$  or  $ER_5$ ,  $U_R = \emptyset$ . After Proposition 3.11, formulæ in  $ER'_1$  and  $ER_5$  are maximal formulæ of  $\mathbb{R}$ .  $\square$

**Corollary 3.13.** GP2, GP3 and GP19 are not maximal to all formulæ in  $\mathbb{R}$ .

*Proof.* Consider the program in Figure 3. The following three test suites can be constructed:

1. For GP2:

Construct a test suite that satisfies the following:  $F = 2$ ,  $P = 9$ ,  $s_8$  ( $s_f$ ) satisfies  $e_f^f = 2 = F \wedge e_p^f = 5 < P$ , and  $s_9$  satisfies  $e_f^g = 1 < F \wedge e_p^g = 4 < e_p^f$ . Then, following the definition of GP2 (which is  $2(e_f + \sqrt{n_p}) + \sqrt{e_p}$ ), the following risk evaluation values are obtained:  $GP2(s_8) = 13$ , which is smaller than  $GP2(s_9) = 14$ . Since  $s_f$  is on  $E$ , this shows that there can exist points outside  $E$  with risk values higher than that of the point on  $E$ . Following Proposition 3.11, GP2 is not the maximal with respect to  $\mathbb{R}^7$ .

2. For GP3:

Construct a test suite that satisfies the following:  $F = 2$ ,  $P = 30$ ,  $s_f$  satisfies  $e_f^f = 2 = F \wedge e_p^f = 25 < P$ , and  $s_9$  satisfies  $e_f^g = 1 < F \wedge e_p^g = 25 = e_p^f$ . Then, following the definition of GP3 (which is  $\sqrt{|e_f^2 - \sqrt{e_p}|}$ ), the following risk evaluation values are obtained:  $GP3(s_f) = 1$ , which is smaller than  $GP3(s_9) = 2$ . Since  $s_f$  is on  $E$ , this shows that there can exist points outside  $E$  with risk values higher than that of the point on  $E$ . Following Proposition 3.11, GP3 is not the maximal to all formulæ in  $\mathbb{R}^8$ .

3. For GP19:

Construct a test suite that satisfies the following:  $F = 5$ ,  $P = 50$ ,  $s_f$  satisfies  $e_f^f = 5 = F \wedge e_p^f = 1 < P$ , and  $s_4$  satisfies  $e_f^4 = 4 < F \wedge e_p^4 < e_p^f = 49 < P$ . Then, following the definition of GP19 (which is  $e_f \sqrt{|e_p - e_f + n_f - n_p|}$ ), the following risk evaluation values are obtained:  $GP19(s_f) = 5\sqrt{5}$ , which is smaller than  $GP19(s_4) = 12\sqrt{5}$ . Since  $s_f$  is on  $E$ , this shows that there can exist points outside  $E$  with risk values higher than that of the point on  $E$ . Following Proposition 3.11, GP19 is not the maximal to all formulæ in  $\mathbb{R}^9$ .  $\square$

With Proposition 3.11, it becomes possible to identify maximal formula with respect to  $\mathbb{R}$ . Furthermore, within these maximal formulæ, we are interested in whether there exists the greatest formulæ and have the following conclusion.

<sup>7</sup>The feasibility of this scenario is analysed in Example 2 of the Appendix.

<sup>8</sup>The feasibility of this scenario is analyzed in Example 3 of the Appendix.

<sup>9</sup>The feasibility of this scenario is analyzed in Example 4 of the Appendix.

### 3.3 Greatest Formulae in $\mathbb{R}$ : The Non-Existence Proof

A greatest formula in  $\mathbb{R}$  is the formula that is better than any other formulae in  $\mathbb{R}$ . It is formally defined as follows:

**Definition 3.14. Greatest Formula.** A risk evaluation formula  $R$  is said to be a *greatest* formula in  $\mathbb{R}$  if, for any formula  $R' \in \mathbb{R} \wedge R' \neq R$ , it holds that  $R \rightarrow R'$ .

Let us now turn to the greatest formula, or, in fact, proving the lack of thereof.

**Proposition 3.15.** There is no formula which is greatest against the set of all formulae,  $\mathbb{R}$ .

*Proof.* Assume that there exists a greatest formula  $R_g$ . Let  $P_{R_g}$  denote the set of  $o_p^{i,j}$  for all pairs of distinct points  $(F, e_p^i)$  and  $(F, e_p^j)$  on the faulty border (where  $e_p^i < e_p^j$ ) and  $U_{R_g}$  denote the sets of points outside the faulty border which have risk values higher than or equal to those of some points  $(F, e_p^i)$  on the faulty border, for formula  $R_g$ . After Proposition 3.11,  $U_{R_g} = \emptyset$ .

Consider the two maximal groups of formulae  $ER'_1$  and  $ER_5$ , which have been proved to be non-equivalent to each other [18]. Let  $U_{ER'_1}$  and  $U_{ER_5}$  denote the sets of points outside the faulty border which have risk values higher than or equal to those of some points  $(F, e_p^i)$  on the faulty border, for formulae in  $ER'_1$  and  $ER_5$ , respectively. Let  $P_{ER'_1}$  and  $P_{ER_5}$  denote the set of  $o_p^{i,j}$  for all pairs of distinct points  $(F, e_p^i)$  and  $(F, e_p^j)$  on the faulty border (where  $e_p^i < e_p^j$ ), for formulae in  $ER'_1$  and  $ER_5$ , respectively. According to Corollary 3.12, it follows that  $U_{ER'_1} = U_{ER_5} = \emptyset$  and  $P_{ER'_1} \neq P_{ER_5}$ . Thus, there are three possible cases for  $P_{R_g}$ , as follows:

- Case 1:  $P_{R_g} = P_{ER'_1}$ . Then it follows that, for  $ER_5$ ,  $U_{ER_5} = U_{R_g} = \emptyset \wedge P_{ER_5} \neq P_{R_g}$ .
- Case 2:  $P_{R_g} = P_{ER_5}$ . Then it follows that, for  $ER'_1$ ,  $U_{ER'_1} = U_{R_g} = \emptyset \wedge P_{ER'_1} \neq P_{R_g}$ .
- Case 3:  $P_{R_g} \neq P_{ER'_1}$  and  $P_{R_g} \neq P_{ER_5}$ . Then it follows that, both for  $ER'_1$  and  $ER_5$ ,  $U_{ER'_1} = U_{R_g} = \emptyset \wedge P_{ER'_1} \neq P_{R_g} \wedge U_{ER_5} = U_{R_g} = \emptyset$  but  $P_{ER_5} \neq P_{R_g}$ .

For any of the above cases, it is possible to construct another formula  $R'$  such that  $U_{R'} = U_{R_g} = \emptyset$  and  $P_{R'} \neq P_{R_g}$ . After Lemma 3.10, we have  $R' \rightarrow R_g$  and  $R_g \rightarrow R'$ . After Definition 3.14,  $R_g$  cannot be the greatest formula.  $\square$

## 4 Visualising the Insights

The spectral coordinate  $\bar{\sigma}$ , introduced in Section 3.1, provides an intuitive way to visualise SBFL formulae. Since neither the size of the test suite nor the exact number of passing and failing test cases is known, we normalise the visualisation with  $P = 100$  and  $F = 100$ . In addition, without affecting generality, we flip the  $P$  axis so that the visualisation depicts  $n_p$  instead of  $e_p$ : this is because, intuitively,  $n_p$  correlates better with the risk value of a statement (the higher  $n_p$  is, the more suspicious the corresponding statement is, because it means that test cases are more likely to pass when not executing the statement). The grids on the plots are separated by the margin of 5.

Figure 4 contains visualisations of Naish2 and GP13, two equivalent formulas<sup>10</sup>. The visualisation provides an intuitive understanding of the equivalence. Both formulas assign any points outside the faulty edge  $E$  with lower risk value than the points on  $E$ . Along  $E$ , the risk value slowly increases monotonically as  $n_p$  increases: therefore, given the same spectrum data, both formulae will rank the statements that belong to  $E$  in the identical order.

<sup>10</sup>An interactive visualisation tool is available at <http://www0.cs.ucl.ac.uk/staff/s.yoo/sbflvis/>.

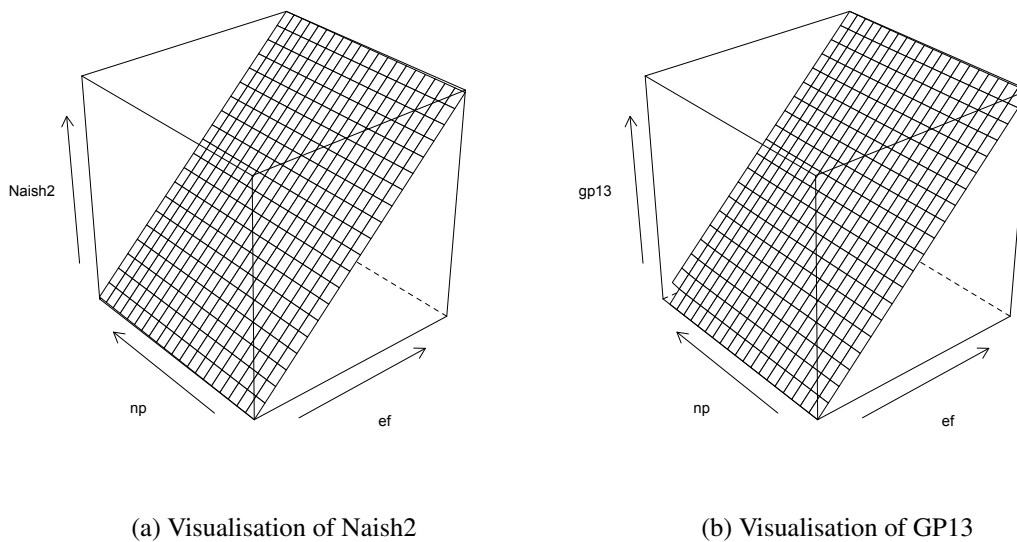


Figure 4: Visualisation of two equivalent SBFL formulæ, Naish2 and GP13

Figure 5 contains visualisations of Tarantula and Jaccard. The Tarantula plot provides an intuitive understanding on why it is not a maximal formula: to many points with  $n_p \approx P$  and  $e_f \approx 0$ , Tarantula assigns significantly high risk values: unless the faulty statement satisfies  $n_p = 0$ , it may be ranked below some of these points.

The visualisation of Jaccard provides another intuitive understanding into a recent observation in SBFL literature. Qi et al. [15] reported that, when used in conjunction with automated bug patching technique GenProg [16], Jaccard outperformed Naish2 by leading GenProg to more generated patches, despite being dominated by Naish2. The fact that Jaccard is dominated by Naish2 can be seen by the fact that some points outside  $E$  can be assigned with higher risk value than other points on  $E$ . However, when the faulty statement has sufficiently high  $n_p$ , the difference in risk values between the faulty statement and others can be much larger with Jaccard than in Naish2 (which only allows very small change of risk value along  $E$ ). Moon et al. introduced a new evaluation metric for SBFL, called Locality Information Loss (LIL), which explained that the bigger difference is more helpful when risk values are used as mutation rates [10]. The visualisation supports this explanation.

## 5 Conclusions and Future Work

Spectrum Based Fault Localisation (SBFL) has received significant amount of attention over the past decade. The focus of the research mainly has been the design of new risk evaluation formulæ that would outperform the existing ones. The evaluation has been of empirical nature, until theoretical analysis began recently. This paper presents the proof that there does not exist the greatest formula, i.e. the one that is better than all other formulæ.

The proof has a significant implication on the research of SBFL. Pursuing the greatest formula is no longer a viable research goal. Concerning SBFL, the future work will be encouraged to consider specialisation, i.e. designing formulæ that are effective in certain contexts, such as a specific project or a particular type of faults. In the wider context, the proof illustrates the limitations of the spectrum based approaches, and encourages fault localisation techniques to consider signals other than program spectrum.

## References

- [1] R. Abreu, P. Zoetewij, and A.J.C. van Gemund. Spectrum-based multiple fault localization. In *Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on, ASE*

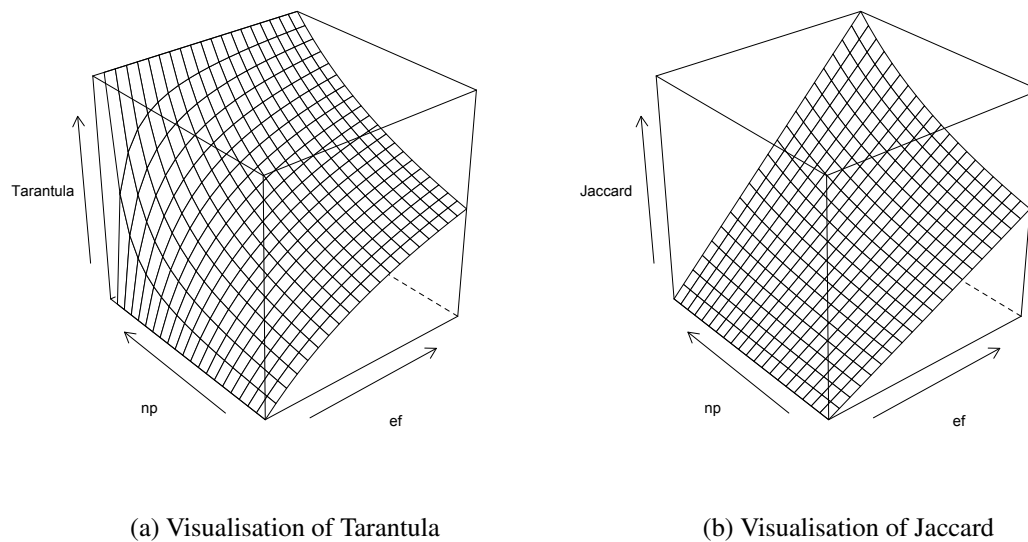


Figure 5: Visualisation of SBFL formulæ: Tarantula and Jaccard

'09, pages 88–99, November 2009.

- [2] Rui Abreu, Peter Zoetewij, and Arjan J. C. van Gemund. On the accuracy of spectrum-based fault localization. In *Proceedings of the Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION*, pages 89–98. IEEE Computer Society, 2007.
- [3] Yanping Chen, Robert L. Probert, and D. Paul Sims. Specification-based regression test selection with risk analysis. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative research (CASCON 2002)*, pages 1–14. IBM Press, October 2002.
- [4] Valentin Dallmeier, Christian Lindig, and Andreas Zeller. Lightweight bug localization with ample. In *Proceedings of the sixth international symposium on Automated analysis-driven debugging, AADEBUG'05*, pages 99–104, New York, NY, USA, 2005. ACM.
- [5] Mary Jean Harrold, Gregg Rothermel, Rui Wu, and Liu Yi. An empirical investigation of program spectra. In *Proceedings of the ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering (PASTE 1998)*, PASTE '98, pages 83–90, New York, NY, USA, 1998. ACM.
- [6] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.
- [7] James A. Jones and Mary Jean Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proceedings of the 20th International Conference on Automated Software Engineering (ASE2005)*, pages 273–282. ACM Press, 2005.
- [8] James A. Jones, Mary Jean Harrold, and John Stasko. Visualization of test information to assist fault localization. In *Proceedings of the 24th International Conference on Software Engineering*, pages 467–477, New York, NY, USA, 2002. ACM.
- [9] James A. Jones, Mary Jean Harrold, and John T. Stasko. Visualization for fault localization. In *Proceedings of ICSE Workshop on Software Visualization*, pages 71–75, 2001.
- [10] Seokhyeon Moon, Yunho Kim, Moonzoo Kim, and Shin Yoo. Ask the mutants: Mutating faulty programs for fault localization. In *Proceedings of the 7th International Conference on Software Testing, Verification and Validation, ICST 2014*, pages 153–162, 2014.



- [11] Lee Naish, Hua Jie Lee, and Kotagiri Ramamohanarao. A model for spectra-based software diagnosis. *ACM Transactions on Software Engineering Methodology*, 20(3):11:1–11:32, August 2011.
- [12] A Ochiai. Zoogeographic studies on the soleoid fishes found in Japan and its neighbouring regions. *Bulletin of the Japanese Society of Scientific Fisheries*, 22(9):526–530, 1957.
- [13] Sangmin Park, Richard W. Vuduc, and Mary Jean Harrold. Falcon: fault localization in concurrent programs. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 245–254, New York, NY, USA, 2010. ACM.
- [14] Chris Parnin and Alessandro Orso. Are automated debugging techniques actually helping programmers? In *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, ISSTA 2011, pages 199–209, New York, NY, USA, 2011. ACM.
- [15] Yuhua Qi, Xiaoguang Mao, Yan Lei, and Chengsong Wang. Using automated program repair for evaluating the effectiveness of fault localization techniques. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, ISSTA 2013, pages 191–201, New York, NY, USA, 2013. ACM.
- [16] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. Automatically finding patches using genetic programming. In *Proceedings of the 31st IEEE International Conference on Software Engineering (ICSE '09)*, pages 364–374, Vancouver, Canada, 16-24 May 2009. IEEE.
- [17] W. Eric Wong, Yu Qi, Lei Zhao, and Kai-Yuan Cai. Effective fault localization using code coverage. In *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 01*, COMPSAC '07, pages 449–456, Washington, DC, USA, 2007. IEEE Computer Society.
- [18] Xiaoyuan Xie, Tsong Yueh Chen, Fei-Ching Kuo, and Baowen Xu. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. *ACM Transactions on Software Engineering Methodology*, 22(4):31:1–31:40, October 2013.
- [19] Xiaoyuan Xie, Fei-Ching Kuo, Tsong Yueh Chen, Shin Chen, and Mark Harman. Provably optimal and human-competitive results in sbse for spectrum based fault localisation. In Günther Ruhe and Yuanyuan Zhang, editors, *Search Based Software Engineering*, volume 8084 of *Lecture Notes in Computer Science*, pages 224–238. Springer Berlin Heidelberg, 2013.
- [20] Shin Yoo. Evolving human competitive spectra-based fault localisation techniques. In Gordon Fraser and Jerffeson Teixeira de Souza, editors, *Search Based Software Engineering*, volume 7515 of *Lecture Notes in Computer Science*, pages 244–258. Springer Berlin Heidelberg, 2012.

## Appendix

### Analysis of the Example Program

In the proof, the program in Figure 3 has been used as an example. The program accepts three independent real numbers:  $x$ ,  $y$  and  $z$ . The statement  $s_8$  contains a fault: the correct predicate should be “**if** ( $x-y < 0$ )”, not “**if** ( $2x-y < 0$ )” and, therefore, is denoted by  $s_f$ . Figure 6 shows the corresponding regions of failure inducing inputs in the space of all possible inputs.

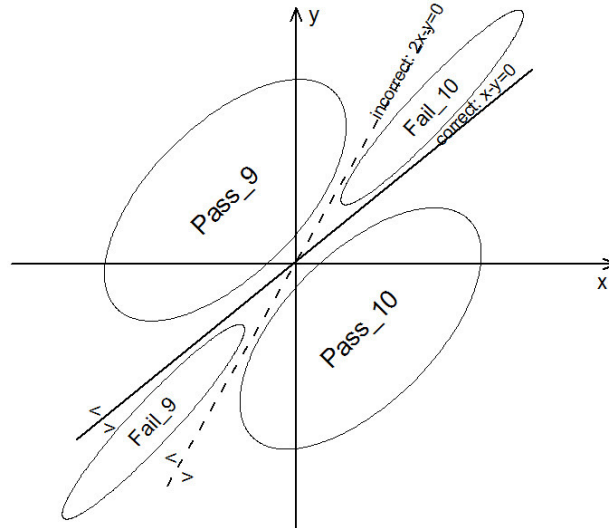


Figure 6: Sample program

First, consider statements  $s_9$  and  $s_{10}$ :

- Any test case  $t_i = (x_i, y_i, z_i)$  such that  $z_i > 7 \wedge (x_i, y_i) \in Fail_9$  will cover  $s_9$  and fail. Let the number of such test cases be  $e_f^9$ .
- Any test case  $t_i = (x_i, y_i, z_i)$  such that  $z_i > 7 \wedge (x_i, y_i) \in Pass_9$  will cover  $s_9$  and pass. Let the number of such test cases be  $e_p^9$ .
- Any test case  $t_i = (x_i, y_i, z_i)$  such that  $z_i > 7 \wedge (x_i, y_i) \in Fail_{10}$  will cover  $s_{10}$  and fail. Let the number of such test cases be  $e_f^{10}$ .
- Any test case  $t_i = (x_i, y_i, z_i)$  such that  $z_i > 7 \wedge (x_i, y_i) \in Pass_{10}$  will cover  $s_{10}$  and pass. Let the number of such test cases be  $e_p^{10}$ .

Since  $s_9$  and  $s_{10}$  are the **true** and **false** branches of the **if** statement in  $s_f$ , only one of these two statements are executed by any test case. Consequently,  $e_f^9 = n_f^{10}$ ,  $n_f^9 = e_f^{10}$ ,  $e_p^9 = n_p^{10}$ , and  $n_p^9 = e_p^{10}$ . There is *no* constraint in selecting test cases from each of the above four regions, and generating test cases from these regions is independent from each other. Therefore, by adjusting the number of test cases in each region, it is possible to have values of  $e_f^9$ ,  $e_p^9$ ,  $e_f^{10}$  and  $e_p^{10}$ .

Next, consider statement  $s_8$  (i.e.  $s_f$ ). It is possible to have any number of passing (i.e.  $e_p^f$ ) and failing (i.e.  $e_f^f = F$ ) test cases by adjusting the above four sets of test cases, because we have  $e_f^9 + e_f^{10} = e_f^f = F$  and  $e_p^9 + e_p^{10} = e_p^f$ .

Then, let us consider statement  $s_7$ . Any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $0 < z_i \leq 7$  will cover  $s_7$  and pass:  $e_p^7$  is equal to the number of such test cases, while  $e_f^7 = 0$ . It is possible to have any number of such test cases. Therefore, by adjusting this number, it is possible to assign any value to  $e_p^7$ .

Now, consider statements  $s_6$ ,  $s_{11}$  and  $s_3$ , whose  $e_f$  values are identical to each other, and so are their  $e_p$ . It can be seen from Figure 3 that  $e_f^6 = e_f^{11} = e_f^3 = e_f^f = F$  and  $e_p^6 = e_p^3 = e_p^{11} = (e_p^f + e_p^7)$ . As a reminder, according to the above analysis,  $e_p^f$  and  $e_p^7$  can be assigned with any values independently.

Next, consider statements  $s_4$  and  $s_5$ . As shown in Figure 3, any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $0 < z_i \leq 12$  will cover  $s_4$ . These test cases can be further categorised as following:

- Any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $0 < z_i \leq 7$  will definitely continue to cover  $s_7$  and thus always pass. The number of these test cases is equal to  $e_p^7$ .
- Any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $7 < z_i \leq 12$  while  $\langle x_i, y_i \rangle \in Pass_9 \cup Pass_{10}$  will also pass. Let us denote the number of these test cases as  $\overline{e_p^4}$ .
- Any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $7 < z_i \leq 12$  while  $\langle x_i, y_i \rangle \in Fail_9 \cup Fail_{10}$  will always fail. The number of these test cases is  $e_f^4$ .

It is not difficult to find that  $e_f^4$  is the size of the subset of failing test cases that cover  $s_f$  and satisfy  $7 < z_i \leq 12$ . Consequently, it follows that  $e_f^4 \leq F$ . On the other hand,  $e_p^4 = e_p^7 + \overline{e_p^4}$ , where  $\overline{e_p^4}$  is the size of the subset of passing test cases that cover and satisfy  $7 < z_i \leq 12$ . Thus, it also follows that  $\overline{e_p^4} \leq e_p^f$ . As a reminder, the values of  $e_p^7$  and  $\overline{e_p^4}$  can be decided independently. Therefore,  $e_p^4$  can be either smaller than, equal to or greater than  $e_p^f$ .

While for  $s_5$ , any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $z_i > 12$  will cover  $s_5$ . These test cases can be further categorised as following:

- Any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $z_i > 12$  while  $\langle x_i, y_i \rangle \in Pass_9 \cup Pass_{10}$  will always pass. The number of these test cases is  $e_p^5$ .
- Any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $z_i > 12$  while  $\langle x_i, y_i \rangle \in Fail_9 \cup Fail_{10}$  will always fail. The number of these test cases is  $e_f^5$ .

Note that  $e_p^5$  is the size of the subset of passing test cases that cover  $s_f$  and satisfy  $z_i > 12$ , while  $e_f^5$  is the size of the subset of failing test cases that cover  $s_f$  and satisfy  $z_i > 12$ . Thus, it follows that  $e_f^5 \leq e_f^f = F$  and  $e_p^5 \leq e_p^f$ .

It should be noted that  $e_f^4$  and  $e_f^5$  are not independent. There is a constraint that  $e_f^4 + e_f^5 = e_f^f = F$ . However, there is no similar constraint on  $e_p^4$  and  $e_p^5$ .

Next, let us consider  $s_2$ . As shown in Figure 3, any test case  $t_i = \langle x_i, y_i, z_i \rangle$  such that  $z_i \leq 0$  will cover  $s_2$  and pass:  $e_p^2$  is equal to the number of such test cases, while  $e_f^2$  is always 0. It is possible to have any number of such test cases. By adjusting this number, it is possible to assign any value to  $e_p^2$ .

Finally, let us consider  $s_1$ . The structure of the program dictates that  $e_f^1 = e_f^f = F$  and  $e_p^1 = (e_p^3 + e_p^2) = (e_p^f + e_p^7 + e_p^2) = P$ . As analyzed above, it is possible to assign, independently, any values to  $e_p^f$ ,  $e_p^7$  and  $e_p^2$ . Consequently,  $e_p^1$  (i.e. the number of total passed test cases  $P$ ) can be any value that is no less than  $e_p^f$ .

### **Feasibility of Test Suites Used in Proofs**

This section presents the analysis of the feasibility of the example test suites used in the proofs.

**Example 1.** With the program in Figure 3, the proof in Proposition 3.11 requires the construction of a test suite such that  $e_p^f$  and  $e_p^4$  are any values, and  $e_f^4 < F$ . According the above discussion, for  $s_f$ , we can assign any value to  $e_p^f$ ; while for  $s_4$ ,  $e_f^4$  can be any value within  $[0, F]$  and  $e_p^4$  can be either smaller than, equal to or greater than  $e_p^f$ . Therefore, it is always possible to construct such a test suite.

**Example 2.** With the program in Figure 3, the proof for GP2 in Corollary 3.13 requires the construction of a test suite such that  $F = 2$ ,  $P = 9$ ,  $s_8$  ( $s_f$ ) has  $e_f^f = 2 = F$  and  $e_p^f = 5 < P$ , and  $s_9$  has  $e_f^9 = 1 < F$  and  $e_p^9 = 4 < e_p^f$ . According to the above analysis, we can assign any value to  $e_p^f$  and any value to  $P$  that is no less than  $e_p^f$ . And for  $s_9$ , we can have any value of  $e_f^9$  within  $[0, F]$  and any value of  $e_p^9$  within  $[0, e_p^f]$ . Therefore, it is always possible to construct such a test suite.

**Example 3.** With the program in Figure 3, the proof for GP3 in Corollary 3.13 requires the construction of a test suite such that  $F = 2$ ,  $P = 30$ ,  $s_f$  has  $e_f^f = 2 = F$  and  $e_p^f = 25 < P$ , and  $s_9$  has  $e_f^9 = 1 < F$  and  $e_p^9 = 25 = e_p^f$ . Similar to the analysis in Example 2, it is always possible to construct such a test suite.

**Example 4.** With the program in Figure 3, the proof for GP19 in Corollary 3.13 requires the construction of a test suite such that  $F = 5$ ,  $P = 50$ ,  $s_f$  has  $e_f^f = 5 = F$  and  $e_p^f = 1 < P$ , and  $s_4$  has  $e_f^4 = 4 < F$  and  $e_p^4 < e_p^f = 49 < P$ . According to the above analysis, for  $s_f$ , we can assign any values to  $e_f^f$  (i.e.  $F$ ) and  $e_p^f$ ; while for  $s_4$ ,  $e_f^4$  can be any value within  $[0, F]$  and  $e_p^4$  can be either smaller than, equal to or greater than  $e_p^f$ ; and  $P$  can be any value that is no less than either  $e_p^4$  or  $e_p^f$ . Therefore, it is always possible to construct such a test suite.

**Example 5.** Let us denote the  $e_p$  values of any two points on the faulty border  $E$  as  $e_p^L$  and  $e_p^H$ , where  $e_p^L < e_p^H$ . For the given program in Figure 3, the proof in Lemma 3.10 requires construction of two test suites, which are referred to as **Test Suite A**, **Test Suite B** and **Test Suite C** in the following discussion.

**Test Suite A:** we have  $e_f^4, e_f^5, e_f^9$  and  $e_f^{10}$  smaller than  $F$ ,  $e_f^1 = e_f^3 = e_f^6 = e_f^f = e_f^{11} = F$ ,  $e_p^f = e_p^L$  and  $e_p^1 = e_p^3 = e_p^6 = e_p^{11} = e_p^H$ . According to the above analysis,  $e_f^4, e_f^5, e_f^9$  and  $e_f^{10}$  can all be less than  $F$  simultaneously. And the  $e_f$  values for  $s_1, s_3, s_6, s_f$  and  $s_{11}$  are always equal to  $F$ . Besides, for  $s_f$ , it is always possible to have any value of  $e_p^f$ ; while it is always possible to have any equal value of  $e_p$  that is larger than  $e_p^f$  for  $s_1, s_3, s_6$  and  $s_{11}$ . Therefore, this test suite is feasible.

**Test Suite B:** we have both  $e_f^4$  and  $e_f^5$  smaller than  $F$ ,  $e_f^1 = e_f^3 = e_f^6 = e_f^f = e_f^9 = e_f^{11} = F$ ,  $e_p^9 = e_p^L$  and  $e_p^1 = e_p^3 = e_p^6 = e_p^f = e_p^{11} = e_p^H$ . As discussed above, it is always possible to have both  $e_f^4$  and  $e_f^5$  smaller than  $F$ . And it is also possible to assign the same value of  $e_f$  (i.e.  $F$ ) to  $s_1, s_3, s_6, s_f, s_9$  and  $s_{11}$ . Moreover,  $s_1, s_3, s_6, s_f$  and  $s_{11}$  are possible to have the same  $e_p$  value that is higher than  $s_9$ . As a consequence, this test suite is always feasible.

**Test Suite C:** we have  $e_f^9$  and  $e_f^{10}$  smaller than  $F$ ,  $e_f^1 = e_f^3 = e_f^4 = e_f^6 = e_f^f = e_f^{11} = F$ ,  $e_p^4 = e_p^L$  and  $e_p^1 = e_p^3 = e_p^6 = e_p^f = e_p^{11} = e_p^H$ . As discussed above, it is always possible to have both  $e_f^9$  and  $e_f^{10}$  smaller than  $F$ . And it is also possible to assign the same value of  $e_f$  (i.e.  $F$ ) to  $s_1, s_3, s_4, s_6, s_f$  and  $s_{11}$ . Moreover,  $s_1, s_3, s_6, s_f$  and  $s_{11}$  are possible to have the same  $e_p$  value that is higher than  $s_4$ . As a consequence, this test suite is always feasible.