METRIC*: A Metamorphic Relation Identification Technique Based on Input and Output Domains

Chang-ai Sun *, *Senior Member, IEEE*, An Fu, Pak-Lok Poon, *Member, IEEE*
Xiaoyuan Xie, *Member, IEEE*, Huai Liu, *Member, IEEE*, and Tsong Yueh Chen, *Member, IEEE*

Abstract—Metamorphic testing is well known for its ability to alleviate the oracle problem in software testing. The main idea of metamorphic testing is to test a software system by checking whether each identified metamorphic relation (MR) holds among several executions. In this regard, identifying MRs is an essential task in metamorphic testing. In view of the importance of this identification task, METRIC (METamorphic Relation Identification based on Category-choice framework) was developed to help software testers identify MRs from a given set of complete test frames. However, during MR identification, METRIC primarily focuses on the input domain without sufficient attention given to the output domain, thereby hindering the effectiveness of METRIC. Inspired by this problem, we have extended METRIC into METRIC* by incorporating the information derived from the output domain for MR identification. A tool implementing METRIC* has also been developed. Two rounds of experiments, involving four real-life specifications, have been conducted to evaluate the effectiveness and efficiency of METRIC*. The results have confirmed that METRIC* is highly effective and efficient in MR identification. Additional experiments have been performed to compare the fault detection capability of the MRs generated by METRIC* and those by μMT (another MR identification technique). The comparison results have confirmed that the MRs generated by METRIC* are highly effective in fault detection.

Index Terms—Metamorphic testing, metamorphic relation, category-choice framework, fault detection effectiveness.

1 Introduction

SOFTWARE testing is an important technique to guarantee the quality of software systems. When quality of software systems refers to the correctness of functions, we test whether a software system can do the right thing in a right way. In general, a software system is considered to *pass* a test if it behaves as expected. Otherwise, it is considered to *fail* a test. A common approach to determine whether a test has passed or failed is to compare the actual output with the expected output. The mechanism that validates the actual output is called an *oracle* [1]. For many software testing techniques, it is assumed that an oracle of the software system under test exists. But, under certain circumstances, an oracle does not exist or is infeasible to apply, which is so called the *oracle problem* [2]. When the oracle problem exists, a testing technique requiring oracles is hard to be applied.

To address the oracle problem in testing, researchers have proposed various techniques, such as assertion [3], N-version programming [4], and metamorphic testing (MT) [5]. Among them, MT is a testing technique which makes use of the characteristics or properties of a software system to detect faults. For a given system, we can derive some essential properties that reflect the inner relationship among several tests according to its specification. These properties are commonly known as *metamorphic relations* (*MRs*) and are expected to be held in the implementation. Otherwise, the implementation is faulty. Because each MR involves more than one test, multiple executions are needed to check whether or not an MR is satisfied. This eliminates the need for comparing the actual output and the expected output for individual test cases. Therefore, the oracle problem is alleviated.

MT is well known for having the following advantages [6]: a) simple but elegant concept: this merit allows even those testers with little experience to learn and apply MT quickly, b) easy automation: given any MRs, their corresponding test case generations and executions, as well as result verifications can be easily automated, and c) low costs: even when the oracle problem exists, the cost of result verification is relatively low (this explains why MT has increasing become popular in different testing scenarios with the oracle problem [2]). Although the above advantages are not unique to MT when compared to assertions and N-version programming, researchers have reported that MT can detect more faults than assertion checking does [7] and the fault detection effectiveness of N-version programming is controversial [8].

Following the increasing popularity of MT, it is reported that, with a small number of "diverse" MRs, MT is sufficient to detect a large number of faults [9]. Segura et al. [10] reviewed recent research of MT and discussed some open challenges associated with the use of MT. Chen et al. [6] summarized research on various aspects of MT and presented several challenges and potential future improve-

* *corresponding author.*

- *C.-A. Sun and A. Fu are with the Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing 100083, China. E-mail: casun@ustb.edu.cn, anfu@xs.ustb.edu.cn*
- *P.-L. Poon is with the School of Engineering and Technology, CQUniversity Australia, Melbourne, Australia. E-mail: p.poon@cqu.edu.au*
- *X. Xie is with the School of Computer Science, Wuhan University, Wuhan, China. E-mail: xxie@whu.edu.cn*
- *H. Liu is with the College of Engineering and Science, Victoria University, Melbourne, Australia. Email: huai.liu@vu.edu.au*
- *T. Y. Chen is with the Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne, Australia. E-mail: tychen@swin.edu.au*

ments on this technique. MT has now been successfully used in various fields including autonomous car driving [11], [12], compiler validation [13], [14], [15], bioinformatics [16], [17], embedded systems [18], [19], scientific and numerical computation [20], encryption programs [21], cybersecurity [22], Web services [23], [24], and RESTful Web APIs [25]. More recently, GraphicsFuzz (a firm established by a team of researchers from Imperial College London, which applies fuzzing and MT for testing graphics drivers across a wide range of mobile and desktop platforms) has been acquired by Google, in response to the growing popularity and well-known effectiveness of MT in fault detection [26], [27].

While MT is receiving more and more attention, some challenging issues still deserve substantial effort to address, such as acquisition of good MRs, effective test case generation, and research on foundation theory of MT. Since MRs play an important role in MT, effective and efficient acquisition of MRs is an essential task. Despite their importance, most MRs are still manually or arbitrarily identified [6]. Furthermore, even experts may have difficulty in identifying MRs [28]. If MRs cannot be identified effectively and efficiently, the application of MT would be limited.

To address this problem, a number of techniques have been proposed such as machine-learning-based MR detection [29], MR composition [30], search-based MR inference [31], graph-kernel-based MR prediction [32], and data-mutation-directed MR acquisition [33]. More recently, Chen et al. [34] developed a systematic and effective MR identification methodology named METRIC (METamorphic Relation Identification based on the Category-choice framework). METRIC makes use of complete test frames constructed by the category-choice framework [35] for MR generation. In METRIC, every two complete test frames form a *candidate pair*, which is then judged by the user whether this pair is useful for deriving an MR.

Among the existing MR identification techniques, METRIC has the following advantages: a) METRIC can be applied to a wide range of application domains while most other MR identification techniques are only applicable to certain domains. For example, the graph-kernel-based MR prediction technique [32] is mainly applicable to scientific programs while METRIC does not have this restriction. b) METRIC has few restrictions or assumptions, which is not common in the existing MR identification techniques. For example, the MR composition technique [30] requires existing MRs to acquire new MRs. c) METRIC considers MR identification at an abstract level rather than at an instance level, thereby making the identification more efficient.

Although the concept of METRIC is simple, intuitive, and elegant, the technique still has some limitations which hinder its effectiveness and efficiency. One limitation is that METRIC does not leverage the information of the output domain for MR identification, which could render the identification process more effective. Another limitation is that METRIC lacks an effective mechanism for reducing the search space of complete test frames when identifying potential MRs, which could be very tedious, time-consuming, and inefficient in those complex scenarios.

This work aims at further improving the effectiveness and efficiency of METRIC by incorporating an output domain-guided mechanism, which not only effectively guides MR identification, but also efficiently reduces the search space during the identification process. Accordingly, we propose an enhanced MR identification technique named METRIC*, which improves METRIC in the following aspects by providing:

a) Systematic guidelines on identifying the fine-coarse relations between the input domain and the output domain, which are useful to increase the effectiveness of MR identification;
b) An output domain-guided mechanism for reducing the search space during MR identification, which enhances the efficiency of identification.

Furthermore, although METRIC is a promising MR identification technique, its fault detection effectiveness has not been verified by means of experiments. In this work, we conducted an empirical study to comprehensively evaluate the fault detection effectiveness of MRs identified via METRIC*, and the improvement of METRIC* against METRIC in terms of the effectiveness and efficiency of MR identification. We also compared the fault detection effectiveness of METRIC* with another identification technique, namely μMT [33]. Experimental results have shown that METRIC* can identify more MRs within a shorter period of time than METRIC. We also found that METRIC* has higher fault detection effectiveness than μMT. To further support the use of METRIC*, we have developed a tool to automate the proposed technique as far as possible.

The main contributions of this paper are:

a) An enhanced MR identification technique named METRIC* is proposed, which includes systematic guidelines on identifying relations from the input domain and the output domain, and a mechanism of reducing the search space during MR identification.
b) A supporting tool named MR-GEN* (MR-GENerator*) is developed, which automates most steps of METRIC* for MR identification.
c) An empirical study involving four real-life specifications with their implementations is conducted to comprehensively evaluate the fault detection effectiveness of METRIC*, and to compare the performance of METRIC* against METRIC in terms of the effectiveness and efficiency of MR identification. We also compare the fault detection effectiveness of MRs identified via METRIC* with those identified via μMT (another MR identification technique).

The rest of this paper is organized as follows: Section 2 introduces the concepts of MT, the category-choice framework and METRIC. Section 3 gives the motivations and the methodology of METRIC*. Section 4 describes the details of an associated tool (MR-GEN*) to support METRIC*. Section 5 describes the settings of our empirical study. Section 6 reports and discusses our empirical results. Section 7 discusses some related work. Section 8 concludes this paper.

2 BACKGROUND

In this section, we revisit MT, the CHOiCe reLATion framEwork (CHOC'LATE) (also called the category-choice framework) [36], and METRIC.

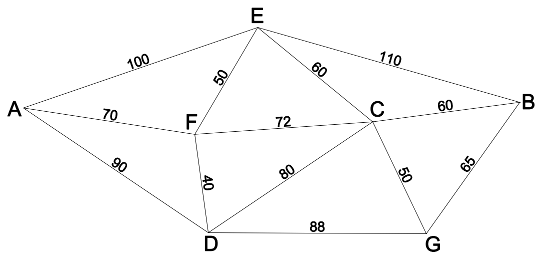


Fig. 1: An undirected graph for example 1

2.1 Metamorphic Testing (MT)

MT was proposed to alleviate the test oracle problem. Its basic steps are as follows:

1) Identify necessary properties of the software under test and derive relations between multiple inputs and their corresponding outputs, commonly known as MRs.
2) Generate test cases using some existing test case generation techniques. These generated test cases are referred to as *source test cases*.
3) Generate *follow-up test cases* from source test cases according to previously defined MRs in Step 1).
4) Execute the software under test with the generated source test cases and their corresponding follow-up test cases, and check whether the MRs in Step 1) hold among the outputs of source test cases and follow-up test cases.

The following example illustrates the details of the above four steps.

Example 1 (Find Shortest Path) Consider a program SP that searches for the shortest path between two specific nodes in a given undirected graph in Fig. 1. We use $SP(G, A, B)$ to represent the shortest path from node A to node B in graph G computed by SP. When graph G is complicated, it is not easy to verify the correctness of the program output. However, there are some properties within SP that we can use for its testing. It is obvious that if we swap the start node and the end node, the length of shortest path between two nodes will not change. Such a property is considered as an MR (denoted by MR_1), as it can be used to derive the variation on output when an input is changed in a specific way. If we construct test cases that satisfy the above differences in terms of input and apply them to SP, the length of output paths should be the same. Otherwise, we have confidence that SP is faulty. Consider a source test case (G, A, B). According to MR_1, a follow-up test case (G, B, A) is constructed. After applying these two test cases to SP, we check whether $|SP(G, A, B)| = |SP(G, B, A)|$. If $|SP(G, A, B)| \neq |SP(G, B, A)|$ and, hence, MR_1 is violated, SP must be faulty.

Another property of SP is that if C is a node in the shortest path between node A and node B, then the length of shortest path from A to B must be the sum of the shortest distance from A to C and that from C to B. The corresponding MR (denoted by MR_2) can be described as follows: $|SP(G, A, B)| = |SP(G, A, C)| + |SP(G, C, B)|$ where C is a node in $SP(G, A, B)$. According to MR_2, we can derive (G, A, C) and (G, C, B) as follow-up test cases from the source test case (G, A, B). After executing these test cases, SP is tested by checking whether or not MR_2 is satisfied.

Example 1 shows that validating the expected output of individual test cases is not required, therefore MT can alleviate the oracle problem. Commonly, an MR consists of two parts: a sub-relation on inputs and a sub-relation on outputs. For instance, in MR_1, the sub-relation on inputs is: the start node and the end node are swapped; the sub-relation on outputs is: the length of the shortest path does not change.

2.2 CHOC'LATE and CHOC'LATE-DIP

CHOC'LATE [35] is a framework that uses categories, choices, and complete test frames to generate test cases from software specifications. The concepts of category, choice and complete test frame are outlined below:

- *Category*: It is a major property or characteristic of an input parameter or an environment condition of the software system that affects its execution behavior.
- *Choice*: The value domain of a category is partitioned into several disjoint sub-domains called choices.
- *Complete test frame*: It is a valid choice combination formed by no more than one choice in each category, from which test cases can be generated.

By selecting a single element from each choice in a complete test frame, this set of elements constitutes a test case. If a complete test frame is homogeneous, any test case selected from it should be a representative value in terms of failure detection. However, it is observed that the same complete test frame may be associated with different output scenarios, which indicates that partitioning the input domain may not yield homogeneous choices [37].

Liu et al. [37] proposed output domain partitioning to further improve the test adequacy of partition testing. With a view to improving the effectiveness of CHOC'LATE, they combined this idea with the category-choice framework and proposed a CHOiCe reLATion framEwork with DIstinguishing outPut scenarios (CHOC'LATE-DIP). The basic concept of CHOC'LATE-DIP is as follows:

- Categories and choices for *output* scenarios are defined using similar approaches as for input parameters and environment conditions. These output-related categories and choices are referred to as *O-categories* and *O-choices*, respectively. The categories and choices related to input parameters and environment conditions are referred to as *I-categories* and *I-choices*, respectively.
- An "extended" choice relation table, including I-choices and O-choices, is constructed, containing the constraint relation for each pair of I-choices, each pair of O-choices, and each pair of an I-choice and an O-choice. Such relations are used to further refine the input partitions.
- "Complete test frames" containing I-choices and O-choices are constructed based on the extended choice relation table. Such "complete test frames" are referred to as *IO-based complete test frames* (*IO-CTFs*).

Note that a "complete test frame" in CHOC'LATE-DIP is different from its counterpart in CHOC'LATE. In CHOC'LATE-DIP, a "complete test frame" (IO-CTF) contains both I-choices and O-choices. In CHOC'LATE, a complete test frame contains I-choices only.

- When generating test cases from an IO-CTF, the type of expected output can be directly determined from the O-choices of that IO-CTF.

As a reminder, an IO-CTF composed of an input choice combination "$\{i_1, i_2, ...\}$" and an output choice combination "$\{o_1, o_2, ...\}$" is denoted as "$\{i_1, i_2, ...; o_1, o_2, ...\}$". We call the input choice combination "$\{i_1, i_2, ...\}$" an *input test frame* and the output choice combination "$\{o_1, o_2, ...\}$" an *output test frame*.

2.3 METRIC

Normally, when identifying an MR, we need to consider how to change an input to another input such that we can define a definite relation among their corresponding outputs. Simply speaking, we need to change an input first and then derive the corresponding predictable change in output. A rather straightforward approach is to change an input directly and then predict the corresponding changes in output by making reference to the specification. This approach is apparently simple but with little guidance. Another approach is to compare different inputs, and predict the corresponding change in terms of output. The latter approach, however, has an essential problem on how to acquire different inputs in the first place. One solution to this problem is to generate different test cases using an existing test case generation method. This solution, however, may involve a large number of generated test cases and, therefore, an excessive number of comparisons has to be performed which could be tedious or even practically infeasible under tight resource constraints.

To alleviate this problem, CHOC'LATE generates complete test frames from which test cases can be constructed. Each complete test frame represents an input scenario and therefore is more abstract than a test case. In contrast to test cases, it is practically more feasible to compare complete test frames. With this merit in mind, Chen et al. [34] proposed their METRIC technique for systematic MR identification, based on the complete test frames generated by CHOC'LATE.

In general, METRIC has the following three main steps:

1) Two distinct complete test frames are selected to form a *candidate pair*.
2) Enable the software tester to determine whether the currently selected candidate pair implies an MR. If so, enable the software tester to describe the inner MR.
3) Repeat from Step 1) until all pairs of complete test frames are examined or the expected number of identified MRs is reached.

In essence, METRIC enables testers to concentrate on only one pair of complete test frames at a time and determine the relationship between the outputs (if any), which is obviously more systematic and efficient than an ad hoc identification approach. As a reminder, candidate pairs that imply MRs are said to be *usable*, and those that do not imply MRs are said to be *unusable*.

3 Motivation and Methodology

This section introduces the motivation and methodology of METRIC* with an example.

3.1 Motivation

CHOC'LATE is a partition testing technique, which involves dividing all possible program inputs into disjoint partitions, from which test cases are selected. The idea is that all test cases within a partition have a good chance to be homogeneous and, hence, a single test case is likely to be sufficient to test that partition. Built upon CHOC'LATE, METRIC provides an effective and practical methodology for MR identification. However, METRIC still has the following limitations which hinder its effectiveness and efficiency:

a) **The information of the output domain is not leveraged for more effective and efficient MR identification.** Liu et al. [37] observed that output domain partitioning could improve the effectiveness of partition testing. Their observation inspires us to extend METRIC into METRIC* by incorporating the concept of output domain partitioning.
b) **A mechanism for reducing the search space of complete test frames during MR identification is missing.** In METRIC, arbitrarily selecting two distinct complete test frames as a candidate pair for MR identification could be tedious and inefficient when a huge number of complete test frames are involved (although, as mentioned earlier, comparing complete test frames is practically more feasible than comparing test cases). Thus, a search-space reduction mechanism is highly desirable.

To overcome the above limitations, we incorporate an output domain-guided mechanism into METRIC. More specifically, the improvements are as follows:

a) **Providing explicit and systematic guidelines on identifying the relations among inputs and the relations among outputs.** By partitioning the output domain, output test frames are generated and used for grouping input test frames. This provides further guidelines on the identification of relations among outputs. Such guidelines significantly improves the effectiveness of MR identification.
b) **Providing a search-space reduction mechanism of complete test frames during MR identification.** The variations of output test frames serve as a clue to help testers differentiate usable candidate pairs from unusable ones. Consequently, the efficiency of MR identification is improved.

3.2 Methodology

Figure 2 outlines the major steps of METRIC*, which are further explained below:

1) Construct a set of IO-CTFs using CHOC'LATE-DIP [37].

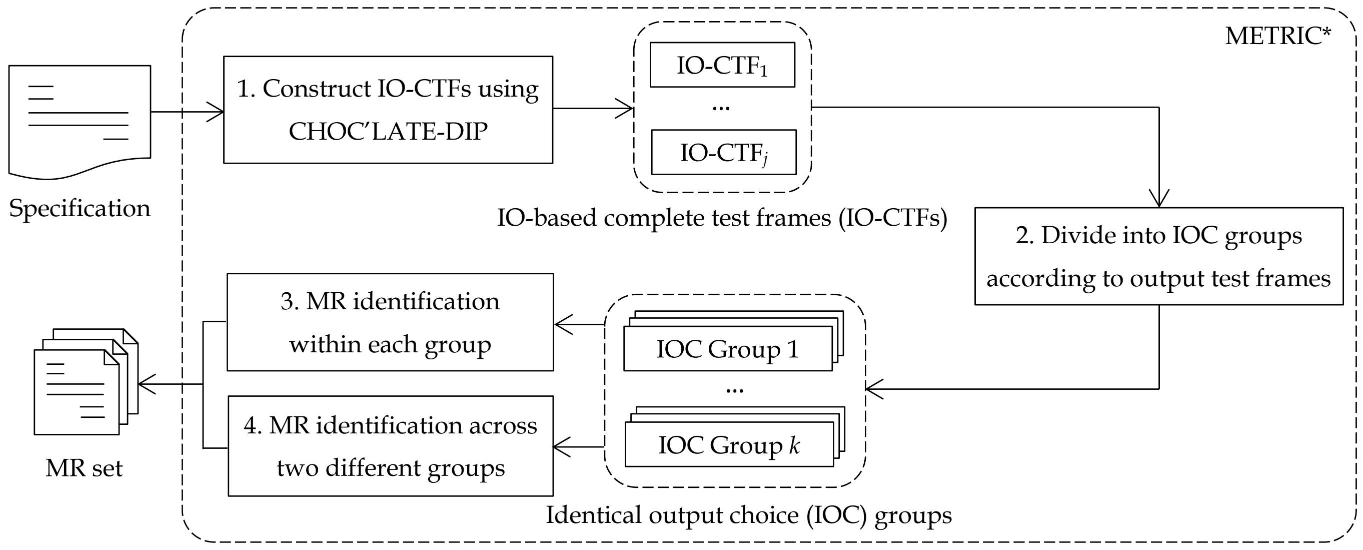

METRIC*
Specification
1. Construct IO-CTFs using CHOC'LATE-DIP
IO-CTF$_1$
...
IO-CTF$_j$
IO-based complete test frames (IO-CTFs)
2. Divide into IOC groups according to output test frames
3. MR identification within each group
4. MR identification across two different groups
IOC Group 1
...
IOC Group k
Identical output choice (IOC) groups
MR set

Fig. 2: Overview of METRIC*

2) Classify all the IO-CTFs into *identical output choice groups*, such that every IO-CTF in the same group has the same output test frame.
3) Identify MRs (if any) from candidate pairs [1] formed by two IO-CTFs *within* each identical output choice group as follows: Select an identical output choice group with two or more IO-CTFs. For the selected group, pick any candidate pair and identify from this pair: (a) any possible sub-relation on the output test frame (denoted by R_o), and (b) any possible sub-relation on the input test frames (denoted by R_i). Thereafter, an MR, comprising R_i and R_o, is formed. Repeat the above process until all candidate pairs have been examined or the expected number of MRs required by the tester has been reached.
4) Identify MRs (if any) from candidate pairs formed by two IO-CTFs *across* two different identical output choice groups as follows: Select two different identical output choice groups. For the selected groups, pick any candidate pair (one IO-CTF from each group) and identify from this pair: (a) any possible sub-relation on the output frames (denoted by R_o), and (b) any possible sub-relation on the input test frames (denoted by R_i). Thereafter, an MR, comprising R_i and R_o, is formed. Repeat the above process until all candidate pairs from the two selected groups have been examined or the expected number of MRs required by the tester has been reached.

Note that the order of steps 3 and 4 above is interchangeable. Also, in these two steps, given any candidate pair, the existence of R_o will be determined first before R_i. If R_o does not exist, this candidate pair will not be further evaluated by the tester in order to save manual effort. The rationale is that, if R_o does not exist, this candidate pair is certainly unusable for MR identification, regardless of whether R_i exists.

In Example 2 below, we use both METRIC and METRIC* to generate MRs for a system function, and contrast the two techniques.

Example 2 (Trigonometric Function) Consider a trigonometric function, which accepts an angle value in degree X, and an operation flag F (whose value is either "sine" or "cosine"), and then computes and outputs either the value of sin(X) or cos(X) in accordance with the value of F.

In this example, METRIC (which makes use of complete test frames generated from CHOC'LATE) works as follows:

Step 1: We first define categories for the input domain according to CHOC'LATE. In this case, X and F are defined as categories. Since both sine and cosine functions are periodic, without loss of generality, we only consider the input domain with one period ($X \in [0^\circ, 360^\circ]$).

Step 2: We next partition the input domain associated with each category into choices as shown in Table 1.

Step 3: Complete test frames are generated by combining choices from different categories according to the choice relations. Table 2 shows all the complete test frames generated by CHOC'LATE.

Step 4: MRs are identified from complete test frames as follows:

4.1 We select two distinct complete test frames as a candidate pair, and then determine whether or not the selected pair is usable for MR identification. For instance, we select "{1b, 2a}" and "{1d, 2b}" as a candidate pair, and then determine whether or not this selected pair is usable. For this pair, we can deduce the following MR: "When the range of X changes from $(0^\circ, 90^\circ)$ to $(90^\circ, 180^\circ)$ and F changes from 'sine' to 'cosine', the output value should decrease".

4.2 We repeatedly select any other distinct and unprocessed pairs of complete test frames as candidate pairs, and determine whether or not there exists an MR for the selected pair, until all distinct candidate

1. Note that the term "candidate pair" has slightly different meanings in METRIC and METRIC*. A candidate pair refers to two complete test frames (which involve only I-choices) in METRIC. However, this term refers to two IO-CTFs in METRIC*.

TABLE 1: Categories and Choices for Trigonometric Function

Categories	Associated choices
1. angle (X)	1a. $0°$
	1b. $(0°, 90°)$
	1c. $90°$
	1d. $(90°, 180°)$
	1e. $180°$
	1f. $(180°, 270°)$
	1g. $270°$
	1h. $(270°, 360°)$
	1i. $360°$
2. operation flag (F)	2a. sine
	2b. cosine

TABLE 2: Complete Test Frames for Trigonometric Function

Complete test frames		
{1a, 2a}	{1b, 2a}	{1c, 2a}
{1d, 2a}	{1e, 2a}	{1f, 2a}
{1g, 2a}	{1h, 2a}	{1i, 2a}
{1a, 2b}	{1b, 2b}	{1c, 2b}
{1d, 2b}	{1e, 2b}	{1f, 2b}
{1g, 2b}	{1h, 2b}	{1i, 2b}

TABLE 3: I-categories and I-choices for Trigonometric Function

I-categories	Associated I-choices
I-1. angle (X)	I-1a. $0°$
	I-1b. $(0°, 90°)$
	I-1c. $90°$
	I-1d. $(90°, 180°)$
	I-1e. $180°$
	I-1f. $(180°, 270°)$
	I-1g. $270°$
	I-1h. $(270°, 360°)$
	I-1i. $360°$
I-2. operation flag (F)	I-2a. sine
	I-2b. cosine

TABLE 4: O-categories and O-choices for Trigonometric Function

O-categories	Associated O-choices
O-1. output value (V)	O-1a. 1
	O-1b. $(0, 1)$
	O-1c. 0
	O-1d. $(-1, 0)$
	O-1e. -1

pairs have been processed, or the expected number of MRs required by the tester has been identified.

Now we turn to METRIC* (which makes use of the IO-CTFs generated by CHOC'LATE-DIP). It works as follows:

Step 1: In addition to defining categories of the input domain (I-categories), we also define categories of the output domain (O-categories). In this case, X and F are defined as I-categories and the output value (V) of the trigonometric function is defined as an O-category.

Step 2: I-choices and O-choices are defined for each I-category and O-category, respectively (see Table 3 and Table 4).[2]

Step 3: IO-CTFs are generated by combining the relevant I-choices and O-choices according to the constraint relations among them. Table 5 shows the IO-CTFs so generated.

Step 4: We identify MRs from the generated IO-CTFs as follows:

4.1 We first arrange IO-CTFs into different identical output choice groups (see Table 6).

4.2 We identify MRs *within* each identical output choice group. For example, we select group O-1a (note that this group is associated with the O-choice "1", which is in turn associated with the definite output value "1", as shown in Table 4) and deduce the following R_o: "The output value remains to be '1' ". Suppose we select the IO-CTFs "{I-1c, I-2a; O-1a}" and "{I-1a, I-2b; O-1a}" as a candidate pair (see Table 6). From this selected pair, we deduce the following R_i: "X changes from $90°$ to $0°$ and F changes from 'sine' to 'cosine' ". By combining R_i and R_o, an MR is derived as follows: "When X changes from $90°$ to $0°$ and F changes from 'sine' to 'cosine', the output value will remain to be '1' ". We repeatedly select any distinct and unprocessed pairs of IO-CTFs from group O-1a as candidate pairs, and derive MRs until all combinations have been examined, or the expected number of required MRs has been identified. Repeat the above in this step for other identical output choice groups until all groups have been processed or the expected number of MRs have been identified.

4.3 We identify MRs *across* two identical output choice groups. For instance, we select groups O-1b and O-1d from Table 6, and deduce the following relation on outputs (R_o): "The output value decreases from the range $(0, 1)$ (corresponds to the O-choice O-1b) to the range $(-1, 0)$ (corresponds to the O-choice O-1d)". We select the IO-CTFs "{I-1b, I-2a; O-1b}" and "{I-1d, I-2b; O-1d}" from the identical output choice groups O-1b and O-1d, respectively, as a candidate pair. From this pair, we are able to deduce the following relation on inputs (R_i): "X changes from $(0°, 90°)$ to $(90°, 180°)$ and F changes from 'sine' to 'cosine' ". By combining R_i and R_o, an MR is derived as follows: "When the range of X changes from $(0°, 90°)$ to $(90°, 180°)$ and F changes from 'sine' to 'cosine', the output value should decrease from the range of $(0, 1)$ to the range of $(-1, 0)$". We repeatedly select any IO-CTF from group O-1b and any IO-CTF from group O-1d to form a candidate pair, and identify MRs until all combinations have been examined, or the expected number of required MRs has been identified.

Below, we describe a situation where R_o does not exist and unusable pairs will not be manually examined by the tester. Consider the group O-1b as an example (see Table 6). When the output test frame ("$(0, 1)$") remains unchanged, we cannot determine the exact R_o. Therefore, any candidate pairs associated with the group O-1b are unusable and the tester does not need to spend effort to examine them. This effort-saving, however, does not exist in METRIC because the tester still has to examine these unusable candidate pairs for potential MRs.

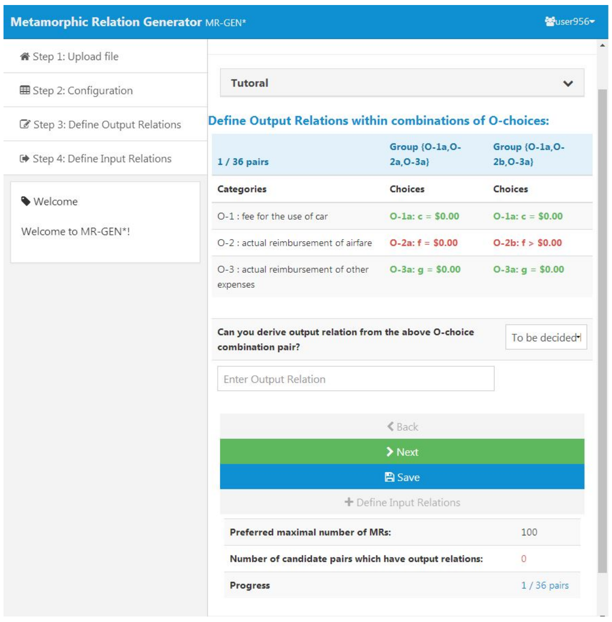
2. As a reminder, Table 3 is effectively the same as Table 1, which is expliclity shown for clarity.

TABLE 5: IO-CTFs for Trigonometric Function

IO-CTFs		
{I-1a, I-2a; O-1c}	{I-1b, I-2a; O-1b}	{I-1c, I-2a; O-1a}
{I-1d, I-2a; O-1b}	{I-1e, I-2a; O-1c}	{I-1f, I-2a; O-1d}
{I-1g, I-2a; O-1e}	{I-1h, I-2a; O-1d}	{I-1i, I-2a; O-1c}
{I-1a, I-2b; O-1a}	{I-1b, I-2b; O-1b}	{I-1c, I-2b; O-1c}
{I-1d, I-2b; O-1d}	{I-1e, I-2b; O-1e}	{I-1f, I-2b; O-1d}
{I-1g, I-2b; O-1c}	{I-1h, I-2b; O-1b}	{I-1i, I-2b; O-1a}

TABLE 6: Identical Output Choice Groups and Their Associated IO-CTFs

Identical output choice groups	Associated IO-CTFs	
O-1a	{I-1c, I-2a; O-1a}	{I-1a, I-2b; O-1a}
	{I-1i, I-2b; O-1a}	
O-1b	{I-1b, I-2a; O-1b}	{I-1d, I-2a; O-1b}
	{I-1b, I-2b; O-1b}	{I-1h, I-2b; O-1b}
O-1c	{I-1a, I-2a; O-1c}	{I-1e, I-2a; O-1c}
	{I-1i, I-2a; O-1c}	{I-1c, I-2b; O-1c}
	{I-1g, I-2b; O-1c}	
O-1d	{I-1f, I-2a; O-1d}	{I-1h, I-2a; O-1d}
	{I-1d, I-2b; O-1d}	{I-1f, I-2b; O-1d}
O-1e	{I-1g, I-2a; O-1e}	{I-1e, I-2b; O-1e}



Define Output Relations within combinations of O-choices:

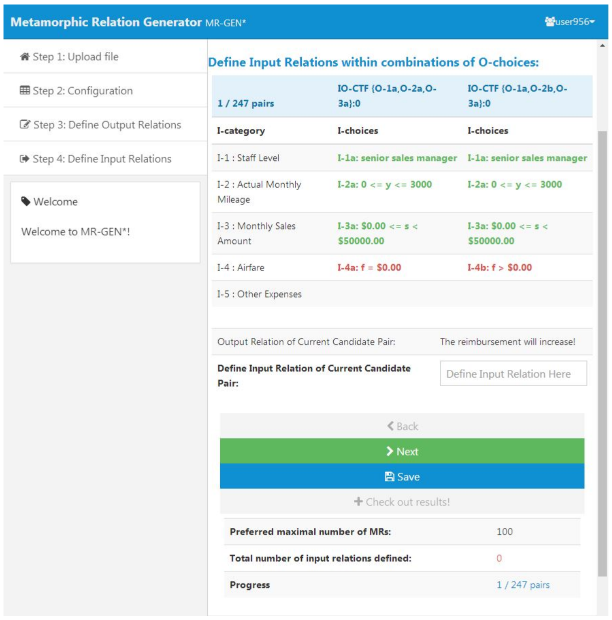
1 / 36 pairs	Group {O-1a,O-2a,O-3a}	Group {O-1a,O-2b,O-3a}
Categories	Choices	Choices
O-1 : fee for the use of car	O-1a: c = $0.00	O-1a: c = $0.00
O-2 : actual reimbursement of airfare	O-2a: f = $0.00	O-2b: f > $0.00
O-3 : actual reimbursement of other expenses	O-3a: g = $0.00	O-3a: g = $0.00

Can you derive output relation from the above O-choice combination pair? To be decided

Enter Output Relation

Preferred maximal number of MRs:	100
Number of candidate pairs which have output relations:	0
Progress	1 / 36 pairs

Fig. 3: Screen for evaluating output test frames



Define Input Relations within combinations of O-choices:

1 / 247 pairs	IO-CTF {O-1a,O-2a,O-3a}:0	IO-CTF {O-1a,O-2b,O-3a}:0
I-category	I-choices	I-choices
I-1 : Staff Level	I-1a: senior sales manager	I-1a: senior sales manager
I-2 : Actual Monthly Mileage	I-2a: 0 <= y <= 3000	I-2a: 0 <= y <= 3000
I-3 : Monthly Sales Amount	I-3a: $0.00 <= s < $50000.00	I-3a: $0.00 <= s < $50000.00
I-4 : Airfare	I-4a: f = $0.00	I-4b: f > $0.00
I-5 : Other Expenses		

Output Relation of Current Candidate Pair: The reimbursement will increase!

Define Input Relation of Current Candidate Pair: Define Input Relation Here

Preferred maximal number of MRs:	100
Total number of input relations defined:	0
Progress	1 / 247 pairs

Fig. 4: Screen for evaluating input test frames

Example 2 above clearly shows that METRIC only considers the derivation of MRs through varying the input test frames (but not the output test frames). By contrast, METRIC* leverages the extra information obtained from output test frames to improve MR identification in two aspects: a) Output test frames are used to group IO-CTFs and further used as a clue to guide MR identification within and across identical output choice groups; and b) The variations of output test frames direct testers to only examine potentially usable pairs for MR identification, and reduce effort wasted on examining unusable pairs.

4 Supporting Tool

This section describes a supporting tool MR-GEN* that we developed (available at the website: http://www.mt4ws.cc/MR-GEN). MR-GEN* supports the following tasks in METRIC*: a) categories, choices, and IO-CTF parsing, b) IO-CTF grouping and comparison, c) automatic removal of unusable pairs for manual evaluation, and d) MR recording. More details of these tasks are provided below:

Step 1: MR-GEN* accepts categories, choices, and IO-CTFs as inputs via a spreadsheet file.

Step 2: Most often, testing resources are limited and, hence, the maximum number of MRs that can be generated and used for testing must be controlled. In this regard, MR-GEN* allows testers to specify the maximum number of MRs to be generated.

Step 3: MR-GEN* groups IO-CTFs into different identical output choice groups. Thereafter, for each unique pair of identical output choice groups, MR-GEN* walks the tester through the relevant IO-CTFs, one pair of IO-CTFs at a time, as shown in Figure 3. MR-GEN* highlights (in red color) the differentiated choices between the two output test frames displayed on-screen to capture the tester's attention. With MR-GEN*, the tester can examine whether the current pair has a relation on outputs (R_o) by selecting a combo box. If an R_o exists, MR-GEN* automatically marks the candidate pairs as "potentially usable" (at this stage, the tester is not completely certain whether the candidate pair under evaluation will eventually turn out to be a usable pair, because the tester has not evaluated the input test frames of this pair) for further consideration. Otherwise, the candidate pairs are marked as unusable and removed. MR-GEN* provides real-time reporting to the tester about the total number of "potentially usable" pairs identified so far. When this number has reached the specified number of MRs to be identified, the tester will proceed to define the relation on inputs for each "potentially usable" pair.

Step 4: MR-GEN* walks the tester through the "potentially usable" pairs, one pair at a time, as shown in Figure 4. Also, the differentiated choices between the two input test frames of the currently selected pair will be highlighted in red color on-screen to capture the tester's attention. MR-GEN* will record the relation on inputs (R_i) (if any) defined by the tester. By combining the relation on inputs and the relation on outputs, METRIC* automatically generates and records an MR for each usable pair. Also, at any time during the identification process, the tester can click the "Check out results" button to review those MRs already defined.

5 Setting of Empirical Study

An empirical study was conducted to compare METRIC* with METRIC with respect to the effectiveness and efficiency of MR identification. Also, the fault detection effectiveness of MRs identified via METRIC* was evaluated and compared with μMT (an instance-level MR identification technique).

5.1 Research Questions

Our empirical study attempted to answer the following research questions:

RQ1: **Can METRIC* be practically used for MR identification?** To answer this question, we conducted experiments involving four commercial software specifications, with a view to determining whether or not METRIC* enables a tester to identify MRs successfully.

RQ2: **Can METRIC* identify more MRs than METRIC?** In this question, we compared the percentages of MRs identified by METRIC* with those by METRIC.

RQ3: **Can METRIC* identify MRs faster than METRIC?** Here, we compared the time spent on identifying a certain number of MRs by METRIC* with that by METRIC.

RQ4: **For those candidate pairs to be manually evaluated by the tester as unusable in METRIC, how many of them can be automatically filtered out by METRIC* to save human effort wasted on evaluation?** We answered this question by calculating the reduction ratio of candidate pairs.

RQ5: **How effective are the MRs identified by METRIC* in fault detection?** We decomposed this question into two sub-research questions. We first evaluated the fault detection effectiveness of MRs identified by METRIC* in terms of mutation scores (*RQ5.1*). We then compared the fault detection effectiveness of MRs identified by METRIC* with those by another MR identification technique μMT (which identifies MRs at the instance level, as opposed to METRIC* which identifies MRs at the input-scenario level) (*RQ5.2*). The reason for choosing μMT for comparison was that both METRIC* and μMT are domain-independent and do not require existing MRs for their applications, while most other techniques either are domain-specific or require existing MRs as their prerequisites.

5.2 Subject Software Systems

Our empirical study involved four subject software systems: Phone Bill Calculation (PHONE), Baggage Billing Service (BAGGAGE), Car and Expense Claim (EXPENSE), and Meal Ordering (MEAL). These systems were implemented using Java programming language.

PHONE is a mobile-phone charge calculation system used by China Unicom. Its purpose is to compute a user's monthly phone charge based on several aspects such as communication time, data usage, and mobile phone tariffs.

BAGGAGE enables passengers to calculate their own baggage fees using the baggage charge scheme of China Airlines. BAGGAGE performs the calculation based on several factors including the relevant flight, baggage, and passenger information (for example, aircraft cabin, flight region, baggage weight, airfare, and eligibility for student discount).

EXPENSE is an expense reimbursement system used in a large trading firm. EXPENSE assists the sales director in determining the fee to be charged to each senior sales manager or sales manager for any "excessive" mileage in the use of the company vehicle, and in processing reimbursement requests regarding various kinds of expenses such as airfare, hotel accommodation, meals, and phone calls. Sales staff members use EXPENSE to make their reimbursement claims at the end of each month.

MEAL is used by an airline catering provider to determine the quantity for every type of meal and other special requests (if any) that need to be prepared and loaded onto the aircraft served by this provider. For each flight, MEAL generates a Meal Schedule Report (MSR), containing the number of various types of meals (first class, business class, economy class, vegetarian, child, crew member, and pilot) and the number of bundles of flowers.

5.3 Category and Choice Identification

For each subject system, categories and choices for its input and output domains were identified in advance based on its corresponding specification. Thereafter, complete test frames and IO-CTFs were generated and separately captured in two Excel files. The set of complete test frames served as the input to METRIC, and the set of IO-CTFs served as the input to METRIC*. To improve the validity of our study, complete test frames and IO-CTFs were provided to the participants for MR identification. This arrangement helped reduce the undesirable impact on the comparison between METRIC and METRIC*, due to the situation whether different sets of complete test frames and IO-CTFs would have been identified by different participants. During the stage of output relation definition, identical output choice groups were randomly selected by MR-GEN* for participants' evaluation. Similarly, during the stage of input relation identification, candidate pairs formed by IO-CTFs were also randomly selected. Such arrangement for random selection reduced the impact of a particular sequence of candidate pairs on our study results. Table 7 provides a summary of the subject specifications and systems, and the numbers of I-categories, I-choices, O-categories, and O-choices involved.

5.4 Participants

Eight participants (denoted by P_1, P_2, ..., P_8) were recruited for our two rounds of experiments related to METRIC and METRIC*. All the participants were postgraduates in

TABLE 7: Total Number of Identified I-categories, I-choices, O-categories, and O-choices

Subject specification	Corresponding subject system	Number of I-categories	Number of I-choices	Number of O-categories	Number of O-choices
S_{PHONE}	PHONE	4	12	2	8
S_{BAGGAGE}	BAGGAGE	5	12	1	2
S_{EXPENSE}	EXPENSE	5	14	3	6
S_{MEAL}	MEAL	7	19	5	15

TABLE 8: Assigned Specifications to Participant Groups (Experiment 1)

Participant groups	Participants	1st Subject specification for MR identification	2nd Subject specification for MR identification
A	P_1, P_2	S_{PHONE}	S_{BAGGAGE}
B	P_3, P_4	S_{EXPENSE}	S_{MEAL}
C	P_5, P_6	S_{BAGGAGE}	S_{PHONE}
D	P_7, P_8	S_{MEAL}	S_{EXPENSE}

TABLE 9: Assigned Specifications to Participant Groups (Experiment 2)

Participant groups	Participants	1st Subject specification for MR identification	2nd Subject specification for MR identification
A	P_1, P_2	S_{EXPENSE}	S_{MEAL}
B	P_3, P_4	S_{PHONE}	S_{BAGGAGE}
C	P_5, P_6	S_{MEAL}	S_{EXPENSE}
D	P_7, P_8	S_{BAGGAGE}	S_{PHONE}

IT with basic knowledge on software testing. These participants were randomly allocated to four groups (A, B, C, D) so that each group had two participants. An individual researcher who is knowledgeable on CHOC'LATE, CHOC'LATE-DIP, MT, METRIC, and METRIC* served as a tutor.

5.5 Measuring the Effectiveness and Efficiency in MR identification

To compare the effectiveness and efficiency of MR identification between METRIC* and METRIC, two rounds of experiments were conducted, involving the following sequence of steps:

- *Tutorial 1*: In this stage, all the recruited participants were firstly taught about the basic concept of MT. Thereafter, they were introduced to CHOC'LATE and its related concepts (category, choice, and complete test frame, in particular) with an example. Furthermore, the participants were introduced to the concept of METRIC and were taught about the functionality of its supporting tool (MR-GEN). Another example was then given to the participants as a hands-on exercise to reinforce their understanding. This stage lasted for about one and a half hours. As soon as this stage had finished, Experiment 1 commenced.
- *Experiment 1*: In this stage, the subject specifications were assigned to the relevant participant groups for MR identification using METRIC, as shown in Table 8. The set of specifications assigned to each participant within a participant group were the same, but were different across participant groups. The two participants within a group were asked to process the two assigned subject specifications in the order as show in Table 8. For example, both participants in Group A were asked to process S_{PHONE} first and then S_{BAGGAGE}. Communications among participants were prohibited during this stage. In other words, each participant performed the identification task independently.

  Note that, candidate pairs were randomly selected by MR-GEN and then provided to a participant, one pair at a time. Each participant was asked to use MR-GEN to independently identify 40 MRs for each specification without time constraint (note that some participants were only able to identify less than 40 MRs even without time constraint). MR-GEN automatically recorded the defined MR and the time spent by a participant on evaluating each candidate pair. The recorded data were used to evaluate the performance of participants with the support of METRIC.
- *Tutorial 2*: In this stage, METRIC* and its associated tool MR-GEN* were introduced to all the participants with an example for illustration. This was followed by another hands-on exercise to reinforce the participants' understanding. Tutorial 2 lasted for about half an hour.
- *Experiment 2*: This stage was conducted after Tutorial 2, during which the subject specifications were assigned to the relevant participant groups for MR identification with the support of METRIC*, in the order as shown in Table 9. Similar to Experiment 1, communication among the participants were prohibited in this stage to ensure that they identified MRs independently. Also, identical output choice groups and candidate pairs were randomly selected and presented to the participants by MR-GEN* to offset (to the extent possible) the impact of a certain sequence of identical output choice groups and candidate pairs on the experimental results. Each participant used MR-GEN* to independently identify 40 MRs for each specification without time constraint. Also, the defined MR and time spent by a participant on evaluating each candidate pair were automatically recorded by MR-GEN*.

5.6 Measuring the Fault Detection Effectiveness

5.6.1 *Mutant Generation*

To evaluate the fault detection effectiveness of METRIC*, mutants for the four subject systems were generated for our experiments using muJava [38]. We used method-level mutation operators to generate mutants; each of which involved only one fault. We generated 210, 187, 180, and 224

TABLE 10: Summary of Generated Mutants

Subject system	Mutation operators used	Number of generated mutants	Number of equivalent mutants
PHONE	AORB, AOIU, AOIS, ROR, COR, COI, LOI, SDL, VDL, ODL	210	38
BAGGAGE	AORB, AOIU, AOIS, ROR, COI, LOI, SDL, VDL, CDL, ODL	187	67
EXPENSE	AORB, AOIU, AOIS, AOUD, ROR, COI, SDL, ODL	180	29
MEAL	AORB, AOIU, AOIS, LOI, SDL, CDL, ODL	224	51

mutants for the subject systems PHONE, BAGGAGE, EXPENSE, and MEAL, respectively. Equivalent mutants were then manually identified and removed from the generated mutants. Table 10 provides further details about the mutation generation process.

5.6.2 Test Case Generation

For each MR generated by METRIC*, its corresponding source test cases and follow-up test cases were generated according to the two IO-CTFs from which this MR was derived. A test case from one IO-CTF served as the source test case and another test case from the other IO-CTF served as the follow-up test case. Source test cases and follow-up test cases were generated from METRIC in a similar way, except that IO-CTFs in METRIC* were replaced by complete test frames in METRIC. Each set of source test cases and its corresponding set of follow-up test case formed a *metamorphic group*.

When generating a source test case from a complete test frame or an IO-CTF, a value was selected from each choice randomly or using the boundary-value approach. On the other hand, when generating a follow-up test case, a value was selected from each choice c based on the following scheme:

- If c was a "common" choice (that is, it also existed in the complete test frame or IO-CTF from which the source test case was generated), the same value used for generating the source test case would be selected from c.
- If c was not a "common" choice, then a value would be selected from c randomly.

In our experiments, we exhausted all candidate pairs for MR identification. Table 11 shows the numbers of test cases generated by the random approach and the boundary-value approach for each subject system.

5.7 Themes of Analysis

5.7.1 Benchmarking Techniques

In this study, we measured the performance of METRIC* by benchmarking it with two other testing techniques: METRIC and μMT. We compared METRIC with METRIC* in terms of the correctness (*RQ2*) and efficiency (*RQ3*) of MR identification. We also evaluated the fault detection effectiveness of MRs identified by METRIC* by comparing with μMT (*RQ5.2*). μMT is a data-mutation directed MR acquisition

TABLE 11: Number of Generated Test Cases

Subject system	Number of generated test cases			
	Random			Boundary value analysis
	1 Test case per MR	5 Test cases per MR	10 Test cases per MR	
PHONE	142	710	1 420	1 096
BAGGAGE	735	3 675	7 350	3 007
EXPENSE	1 130	5 650	11 300	7 772
MEAL	3 512	17 560	35 120	50 907

technique, which uses input data mutation operators and output mapping rules to identify MRs from the software under test. In μMT, data mutation operators are used to define the data mutation relations on the inputs of test cases. Then, each data mutation relation is validated against the given constraints over inputs. Finally, mapping rules of the program are used to define the relations on outputs. Similar to METRIC and METRIC*, μMT is a domain-independent MR acquisition technique with no prerequisite for existing MRs for further identification of new ones. (Unlike μMT, METRIC, and METRIC*, most other MR acquisition techniques are either domain-specific or require existing MRs.) Because of this similarity, METRIC and μMT were used for benchmarking with METRIC*.

5.7.2 Measurement

In our experiments, any MR identified from the participants is called *potential MR*. A potential MR is *genuine* if it correctly reflects the relationship among the inputs and outputs corresponding to a candidate pair. Otherwise, it is *non-genuine*.

For *RQ1*, we used the number of genuine MRs identified by the participants (without time constraint) to measure the practicality of METRIC* for MR identification. For *RQ2*, we used the *correctness ratio* (C) to measure the number of genuine MRs to the number of potential MRs identified by the participants.

For *RQ3*, we measured the time spent (T) for identifying a certain number of genuine MRs by the participants. We also measured the average time spent (T_g) for the participants to identify a genuine MR. In other words, T_g is defined as the ratio of T and the number of genuine MRs identified. It is obvious that higher values of C indicate better results. On the other hand, lower values of T and T_g are preferred because they indicate the participants spent less time on MR identification.

Now we turn to *RQ4*. Let P* and P denote the number of candidate pairs that need to be manually evaluated by the participants in METRIC* and METRIC, respectively. We defined *reduction ratio* (R) as the value of ((P − P*) / P). In this way, R measures how effective METRIC* is in reducing the search space of candidate pairs (and, hence, human evaluation effort) when compared with METRIC. Obviously, higher values of R mean better results.

For *RQ5.1* and *RQ5.2*, we use *mutation score* (MS) to evaluate the fault detection effectiveness of identified MRs. Suppose that a test suite TS generated from an MR set S_MR kills N_k mutants, N denotes the total number of mutants generated from the program under test, and N_eq denotes the number of equivalent mutants. The mutation score of S_MR is calculated as: $MS = N_\mathrm{k} \ / \ (N - N_\mathrm{eq})$. In this way, higher mutation scores mean better results.

TABLE 12: Number of Genuine MRs Identified by Participants

Subject specification	Participants	Number of genuine MRs
S_{PHONE}	P_5	40
	P_6	37
	P_7	40
	P_8	40
	Average	39.3
S_{BAGGAGE}	P_5	40
	P_6	36
	P_7	40
	P_8	40
	Average	39.0
S_{EXPENSE}	P_1	40
	P_2	39
	P_3	32
	P_4	40
	Average	37.8
S_{MEAL}	P_1	32
	P_2	35
	P_3	24
	P_4	40
	Average	32.8

TABLE 13: Average Time (T_g) for Identifying a Genuine MR

Subject specification	METRIC		METRIC*	
	Participants	T_g (sec)	Participants	T_g (sec)
S_{PHONE}	P_1	140	P_5	52
	P_2	99	P_6	80
	P_3	37	P_7	45
	P_4	49	P_8	49
	Average	**81.3**	**Average**	**56.5**
$S_{BAGGAGE}$	P_1	74	P_5	38
	P_2	56	P_6	69
	P_3	66	P_7	53
	P_4	53	P_8	63
	Average	**62.3**	**Average**	**55.8**
$S_{EXPENSE}$	P_5	197	P_1	61
	P_6	217	P_2	37
	P_7	74	P_3	40
	P_8	93	P_4	49
	Average	**145.3**	**Average**	**46.8**
S_{MEAL}	P_5	181	P_1	132
	P_6	229	P_2	90
	P_7	77	P_3	121
	P_8	154	P_4	143
	Average	**160.3**	**Average**	**121.5**

6 Experimental Results and Analyses

6.1 Practicability of METRIC* (*RQ1*)

Without time constraint, the numbers of genuine MRs identified for each subject specification by the relevant participants are given in Table 12. The table shows that, with the support of METRIC*, every participant identified a number of genuine MRs (mean = 37.2; range = [24, 40]) for the assigned specifications, thereby demonstrating the practicality of METRIC* for MR identification.

We further observe that some MRs identified by METRIC* were more precise than their counterpart MRs identified by METRIC. This observation is illustrated by the MR identified from the candidate pair "{1b, 2a}" and "{1d, 2b}" discussed in Step 4.1 of METRIC, and its counterpart MR identified from the candidate pair "{I-1b, I-2a; O-1b}" and "{I-1d, I-2b; O-1d}" in Step 4.2 of METRIC* (see Example 2 in Section 3.2). For the former candidate pair "{1b, 2a}" and "{1d, 2b}" in METRIC, its corresponding MR is: "When the range of X changes from $(0°, 90°)$ to $(90°, 180°)$ and F changes from 'sine' to 'cosine', the output value should decrease". On the other hand, for the latter candidate pair "{I-1b, I-2a; O-1b}" and "{I-1d, I-2b; O-1d}" in METRIC*, its corresponding MR is: "When the range of X changes from $(0°, 90°)$ to $(90°, 180°)$ and F changes from 'sine' to 'cosine', the output value should decrease from the range of $(0, 1)$ to the range of $(-1, 0)$". When comparing the above two MRs, one identified by METRIC and the other by METRIC*, it is obvious that the MR by METRIC* is more precise. It is not difficult to see from Example 2 in Section 3.2 that the reason for the higher preciseness of MR by METRIC* is the provision of additional information about the output test frames in IO-CTFs.

6.2 Correctness of MR Identification (*RQ2*)

To answer *RQ2*, we compared the performance of the participants supported by METRIC and METRIC*, in terms of correctness ratios (as defined in Section 5.7.2).

Figures 5 to 8 show the study results, with each figure corresponding to one subject specification. In this part of the experiments, the eight participants were divided into two groups. Group 1 consisted of participants P_1, P_2, P_3, and P_4; whereas Group 2 consisted of participants P_5, P_6, P_7, and P_8. Group 1 was asked to identify MRs from S_{PHONE} and $S_{BAGGAGE}$ using METRIC, and to identify MRs from $S_{EXPENSE}$ and S_{MEAL} using METRIC*. Similarly, Group 2 was asked to identify MRs from $S_{EXPENSE}$ and S_{MEAL} using METRIC, and to identify MRs from S_{PHONE} and $S_{BAGGAGE}$ using METRIC*. This arrangement allowed us to compare the effectiveness of METRIC and METRIC*, and at the same time reducing the impact of the participants' backgrounds and credentials on the study results.

Across all the four figures, on average, the correctness ratios involving the use of METRIC* (mean = 0.930, range = [0.819, 0.981]) were higher than those involving METRIC (mean = 0.810, range = [0.681, 0.919]). If we consider the specifications individually, on average, the increases in correctness ratios from using METRIC to METRIC* were 12.9%, 6.1%, 22.8%, and 20.3% (mean = 15.5%) for S_{PHONE}, $S_{BAGGAGE}$, $S_{EXPENSE}$, and S_{MEAL}, respectively. This shows that, when compared with METRIC, METRIC* helped the participants identify more genuine MRs.

One plausible reason contributing to the higher correctness ratios associated with METRIC* (when compared with METRIC) is the provision of additional information derived from the output test frames. Given a candidate pair in METRIC*, the variation of output from one input test frame to another input test frame is explicitly and easily derived by comparing the output test frames of a candidate pair, which greatly eases MR identification. This advantage, however, does not exist in METRIC because this technique requires the user to additionally derive the type of outputs associated with each complete test frame in a candidate pair under evaluation. Obviously, this derivation requires time and effort to complete, and it may involve human mistake.

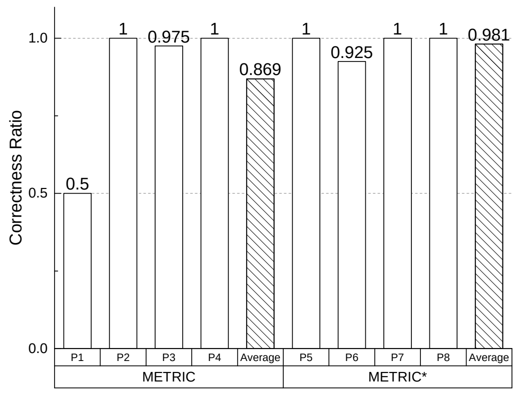


Fig. 5: Correctness ratio ($\mathrm{S}_{\mathrm{PHONE}}$)

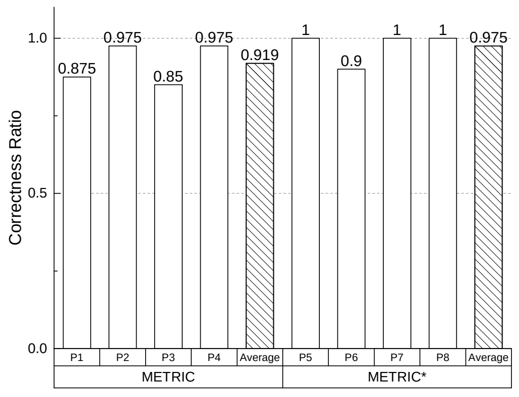


Fig. 6: Correctness ratio ($\mathrm{S}_{\mathrm{BAGGAGE}}$)

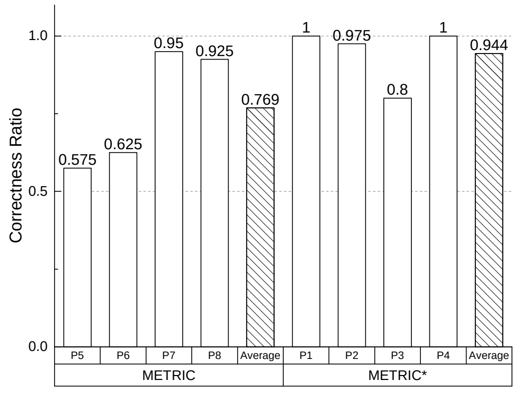


Fig. 7: Correctness ratio ($\mathrm{S}_{\mathrm{EXPENSE}}$)

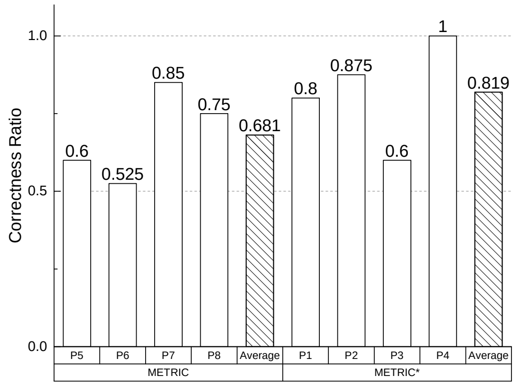


Fig. 8: Correctness ratio ($\mathrm{S}_{\mathrm{MEAL}}$)

6.3 Efficiency of MR Identification (*RQ3*)

To address *RQ3*, we measured the average time spent (T_{g}) by each participant for identifying a genuine MR with the support of METRIC or METRIC*. Table 13 shows the results. It can be seen from the table that, across all the four subject specifications, the average values of T_{g} for METRIC* were much lower than those for METRIC. For example, considering all the four specifications together, the average values of T_{g} for METRIC and METRIC* were 112.3 seconds and 70.1 seconds, respectively. In other words, the value of T_{g} was decreased by about 37.5% from METRIC to METRIC*. This was an encouraging result, indicating that METRIC* enabled the participants to identify genuine MRs much faster when compared with METRIC.

The higher identification efficiency of METRIC* are mainly attributed to the following two reasons:

- Pre-filtering of some unusable pairs before presenting candidate pairs to the user for evaluation: This arrangement generally increases the percentage of usable pairs in candidate pairs for user evaluation. Consequently, the "non-productive" time spent wasted on evaluating candidate pairs which turn out to be unusable is reduced.
- The provision of additional information derived from the output test frames of the candidate pairs: As explained in Section 6.2 above, the additional information makes the task of MR identification faster and more accurate.

In addition to T_{g}, with respect to the first 20 genuine MRs identified by the participants, we also measured the change in the average time spent T_{x}^{i} (across all the participants) on identifying an *extra* ith genuine MR over time, as more and more genuine MRs have been identified. We chose 20 genuine MRs because of two observations: (a) one of the eight participants had only identified a total of 23 genuine MRs (without time constraint), and (b) the time spent for identifying an extra genuine MR varied little after the first 20 had been identified (across all the participants).

To compute the average value of T_{x}^{i} for METRIC, the method was straightforward and obvious and, hence, needs no further explanation. For METRIC*, however, it deserves some discussion. In METRIC*, given a candidate pair, MR identification involves two sub-tasks: (a) identifying the relation between the input test frames of the candidate pair, and (b) identifying the relation between the output test frames of the candidate pair. Let T_{in}^{i} and T_{out}^{i} denote the time spent on sub-tasks (a) and (b) for the ith genuine MR, respectively. Suppose:

- a metamorphic relation MR_i was formed from the candidate pair IO-CTF_1 and IO-CTF_2, which are from either the same identical output choice group or two different identical output choice groups;

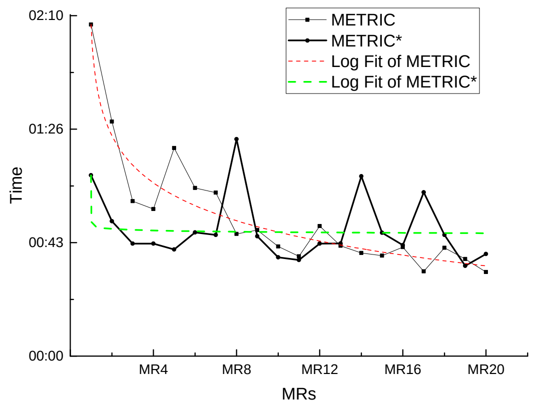


Fig. 9: Average time spent (T_x^i, where $1 \leq i \leq 20$) for identifying each of the first 20 genuine MRs (S_{PHONE})

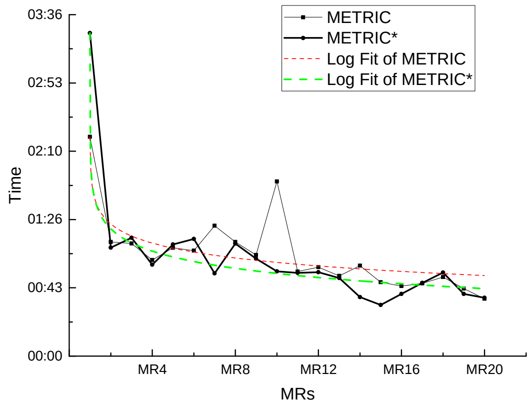


Fig. 10: Average time spent (T_x^i, where $1 \leq i \leq 20$) for identifying each of the first 20 genuine MRs (S_{BAGGAGE})

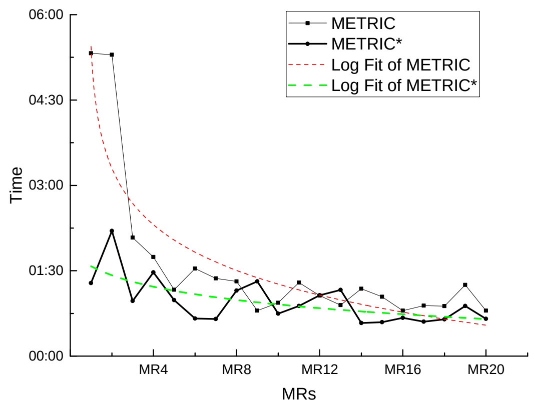


Fig. 11: Average time spent (T_x^i, where $1 \leq i \leq 20$) for identifying each of the first 20 genuine MRs (S_{EXPENSE})

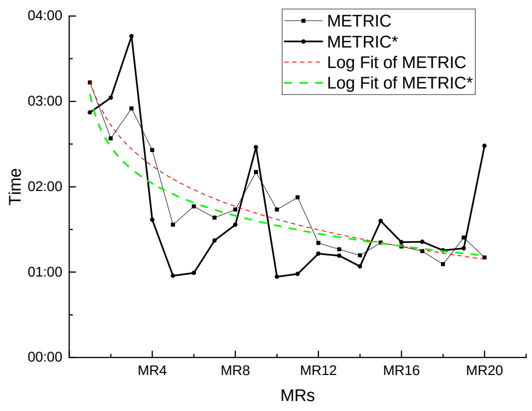


Fig. 12: Average time spent (T_x^i, where $1 \leq i \leq 20$) for identifying each of the first 20 genuine MRs (S_{MEAL})

- the amount of time T_{m}^i was spent by a participant P for identifying the relation between the input test frames of IO-CTF$_1$ and IO-CTF$_2$;
- the amount of time T_{n}^i was spent by a participant P for identifying the relation between the output test frames of IO-CTF$_1$ and IO-CTF$_2$; and
- there are k usable pairs in all the candidate pairs.

Obviously, $T_{\text{in}}^i = T_{\text{m}}^i$. Also, $T_{\text{out}}^i = T_{\text{n}}^i/k$. This was because once the relation between the output test frames of IO-CTF$_1$ and IO-CTF$_2$ was manually defined by the tester, this relation would be automatically recorded by MR-GEN* and applied to all other associated candidate pairs. In view of this, T_{n}^i should be evenly distributed among the k usable pairs. Furthermore, the time spent (T_{x}^i) by participant P on defining the ith genuine MR is the sum of T_{in}^i and T_{out}^i.

Figures 9 to 12 show the change in T_{x}^i over time for the four subject specifications, as more and more genuine MRs have been identified. In each of these four figures, the x-axis indicates the ith (where $1 \leq i \leq 20$) genuine MR identified, and the y-axis indicates the values of T_{x}^i for these MRs. In addition to the two line charts for METRIC (the thin solid line) and METRIC* (the thick solid line), their corresponding logarithmic fits (using the regression formula: $y = b \ln{(x + c)}$ where a, b, and c are parameters) are also shown in the figures for analysis.

We have the following observations from the curves in these four figures:

- For both METRIC and METRIC* and for all the four specifications, the values of T_{x}^i generally decreased as the number of identified genuine MRs increased.
- All the participants spent significantly more time identifying the first two genuine MRs than the latter ones.

The above observation are well explained by means of the *learning curve effect* — the more times a task (e.g., MR identification) has been performed, the less time is required on each subsequent iteration. In our experiments, as the identification task progressed, the participants were becoming more familiar with the subject specifications and both identification techniques (METRIC and METRIC*).

We further note an observation from the two logarithmic fits for S_{PHONE} (Figure 9), S_{EXPENSE} (Figure 11), and S_{MEAL} (Figure 12). For these three specifications, the fits in METRIC drop at a faster rate than those in METRIC*, indicating that the learning curve effect (and, hence, the decrease in T_{x}^i as the number of identified genuine MRs increased) was more profound for METRIC than METRIC*. This can be explained by the provision of the output test frames in METRIC* (but not METRIC). In METRIC, whenever participants evaluated a given candidate pair, they need to spend time on deducing

TABLE 14: Reduction Ratios of Candidate Pairs

Subject specification	Numbers of candidate pairs		Reduction ratio (R)
	P* (METRIC*)	P (METRIC)	
S_{PHONE}	142	496	71.4%
$\text{S}_{\text{BAGGAGE}}$	735	780	5.8%
$\text{S}_{\text{EXPENSE}}$	1 130	2 145	47.3%
S_{MEAL}	16 110	16 110	0.0%

TABLE 15: Average Mutation Scores of Different Test Suites

Subject system	Average mutation scores (10 rounds of analyses)			
	Random approach			Boundary value analysis
	1 test case per MR	5 test cases per MR	10 test cases per MR	
PHONE	69.3%	74.7%	76.6%	80.7%
BAGGAGE	91.7%	91.7%	91.7%	91.7%
EXPENSE	75.8%	76.6%	77.2%	86.1%
MEAL	74.6%	74.6%	74.6%	74.6%

the variation on outputs caused by a variation on inputs. Due to the learning curve effect, this time spent would generally decrease as the MR identification task proceeded. However, in METRIC*, such decrease in time spent was relatively smaller, because participants could easily deduce the variation of outputs by referring to the output test frames of the candidate pair (less reliance on the participants' prior identification experience), thereby diminishing the learning curve effect.

We note an exception to the above trend of the logarithmic fits in $\text{S}_{\text{BAGGAGE}}$ (Figure 10). Here, the fits in both METRIC and METRIC* are about the same. A close examination revealed a plausible reason: identifying output sub-relations for $\text{S}_{\text{BAGGAGE}}$ was simpler than for the other three subject specifications. This was because $\text{S}_{\text{BAGGAGE}}$ only involved 1 O-category and 2 O-choices. These numbers contrasted with the large numbers of O-categories and/or O-choices for the other three specifications (S_{PHONE} involved 2 O-categories and 8 O-choices, $\text{S}_{\text{EXPENSE}}$ involved 3 O-categories and 6 O-choices, and S_{MEAL} involved 5 O-categories and 15 O-choices). Because $\text{S}_{\text{BAGGAGE}}$ only involved a small number of O-categories and O-choices, the learning curve effect when applying METRIC (without the merit of having additional information from output test frames) to $\text{S}_{\text{BAGGAGE}}$ was not significant. This resulted in similar logarithmic fits for $\text{S}_{\text{BAGGAGE}}$ in METRIC and METRIC*.

6.4 Reduction in Candidate Pairs for Manual Evaluation (*RQ4*)

RQ4 is answered by calculating the reduction ratio ($R = (P - P\text{*}) \ / \ P$) of candidate pairs. Results are shown in Table 14.

Except S_{MEAL}, it was observed that the number of candidate pairs to be manually evaluated by the participants has been reduced with the support of METRIC*. Considering the three subject specifications S_{PHONE}, $\text{S}_{\text{BAGGAGE}}$, and $\text{S}_{\text{EXPENSE}}$ together, on average, the value of R was 41.5%. Undoubtedly, this extent of reduction was fairly significant, indicating that a great deal of manual effort in evaluating candidate pairs for MR identification has been saved.

The zero value of R for S_{MEAL} warrants detailed investigation. We found that there was only one IO-CTF in each identical output choice group for S_{MEAL}. In this case, Step 3 of METRIC* (see Section 3.2) has become inapplicable. Thus, the merit of reducing the number of candidate pairs (IO-CTFs) associated with the same identical output choice group for manual evaluation did not exist. On the other hand, the number of candidate pairs (in which each IO-CTF is selected from a different identical output choice group) to be manually evaluated in METRIC* is the same as the number of candidate pairs for manual evaluation in METRIC. The above explains the zero value of R for S_{MEAL}. Thus, in the worst scenario (that is, the number of IO-CTFs in each identical output choice group is one), the number of candidate pairs related to this group to be manually evaluated in METRIC* is the same as the number of their counterpart candidate pairs in METRIC.

TABLE 16: Characteristics of Non-killed Mutants

Subject system	Mutation operators	Number of non-killed mutants	Percentage of non-killed mutants
PHONE	AOIS	12	40.0%
	AORB	8	26.7%
	ODL	2	6.7%
	SDL	8	26.7%
	VDL	2	6.7%
BAGGAGE	AOIS	1	10.0%
	AORB	4	40.0%
	CDL	3	30.0%
	SDL	3	30.0%
EXPENSE	AOIS	15	71.4%
	ROR	5	23.8%
	SDL	1	4.8%
MEAL	AOIS	14	32.6%
	AORB	15	34.9%
	CDL	6	14.0%
	ODL	6	14.0%
	SDL	2	4.7%

6.5 Fault Detection Effectiveness of MRs (*RQ5.1*)

This part of the study involved 10 rounds of mutation analyses in order to make the results more credible. In each round, we constructed four test suites for each subject system involving METRIC*. Among these four test suites, three were constructed randomly by generating 1, 5, and 10 test cases from each MR generated by METRIC*, respectively (hence, these three test suites were of different sizes). On the other hand, the remaining test suite was constructed using boundary value analysis on the MRs generated by METRIC*. In this scheme, a total of 160 ($= 10 \times 4 \times 4$) test suites were constructed. For each of these test suites, we computed its mutation score (which was an indicator of that test suite's fault detection effectiveness; see Section 5.7.2 for more details). Table 15 shows the average mutation scores for each subject system.

Considering all the test suites generated randomly or by boundary value analysis, across all the four subject systems, the average mutation scores ranged from 69.3% to 91.7%, with a mean of 80.1%. This promising observation indicated that the high fault detection effectiveness of the MRs generated by METRIC* (Observation 1). It was also interesting to observe how the average mutation scores varied with the size of a test suite. For the subject systems BAGGAGE and MEAL, the average mutation scores remained unchanged as the size of the test suites grew. On the other hand, for

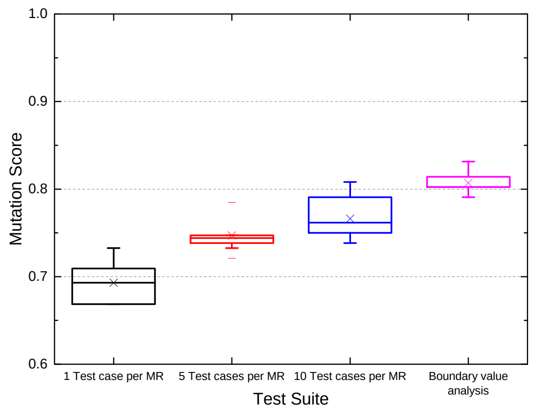


Fig. 13: Mutation score distributions of PHONE (METRIC*)

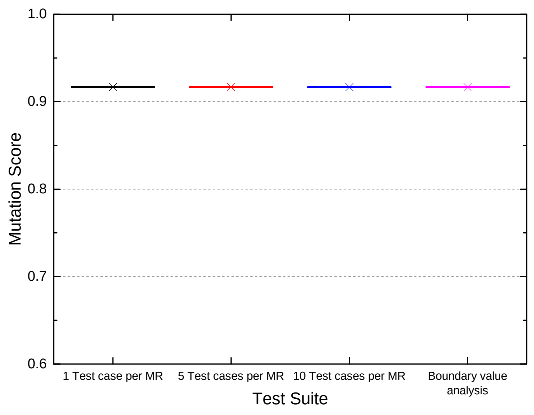


Fig. 14: Mutation score distributions of BAGGAGE (METRIC*)

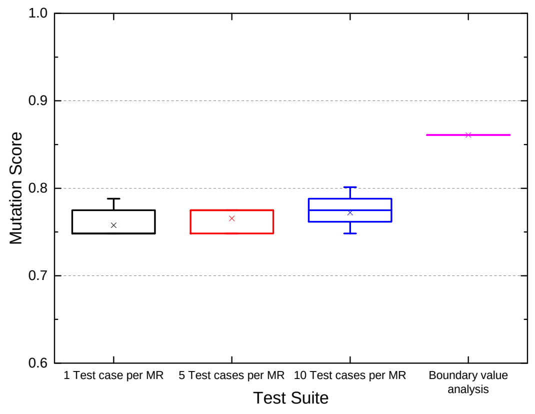


Fig. 15: Mutation score distributions of EXPENSE (METRIC*)

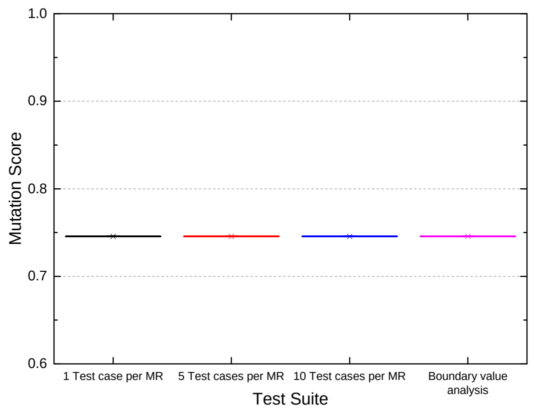


Fig. 16: Mutation score distributions of MEAL (METRIC*)

the subject systems PHONE and EXPENSE, the average mutation scores only showed a small increase even when the size of a test suite grew significantly. For example, for PHONE, even when the size of the test suite grew 10 times larger (from 1 test case to 10 test cases generated per MR), the average mutation scores only showed an increase of 7.3% (= 76.6% − 69.3%). Again, this was an encouraging observation (Observation 2). If we consider the above two observations together, it means that we do not need to spend a lot of effort in generating and executing a *large* test suite for testing (via METRIC*), but at the same time can achieve a fairly higher fault detection effectiveness. Plausible reasons for these encouraging observations are given as follows. Intuitively, an IO-CTF (and also a complete test frame) corresponds to a partition in partition testing. Thus, test cases constructed from the same IO-CTF are likely to be related to the same fault if they are failure-causing and, hence, have similar fault detection effectiveness. Following this argument, generating more test cases from the same MR identified from the same IO-CTF should not largely increase the number of faults detected (Observation 2). Also, different IO-CTFs are obviously "heterogeneous". In this regard, the approach of generating MRs from various IO-CTFs will result in a set of *diverse* MRs. It is reported in [9] that even a small number of *diverse* MRs can achieve high fault detection effectiveness (Observation 1).

The mutation score distributions of different test suites generated for the four subject systems are shown as box plots in Figures 13 to 16. All these box plots show that the distributions of mutation scores among the four systems (in particular, BAGGAGE and MEAL) were confined to narrow regions. This observation indicates that the fault detection effectiveness of MRs identified by METRIC* is relatively stable without large variation.

We have also analyzed the type of mutants that were least likely to be killed by the generated test suites. The relevant statistics are shown in Table 16. In this table, it was observed that the mutants generated by mutation operators AOIS and AORB were least likely to be killed by the generated test suites. In other words, the MRs generated by METRIC* were relatively less effective in detecting faults generated by these two mutation operators.

TABLE 17: Number of MRs Identified by μMT and METRIC*

Subject system	Number of MRs identified by	
	μMT	METRIC*
PHONE	32	142
BAGGAGE	36	735
EXPENSE	60	1 130
MEAL	80	3 152

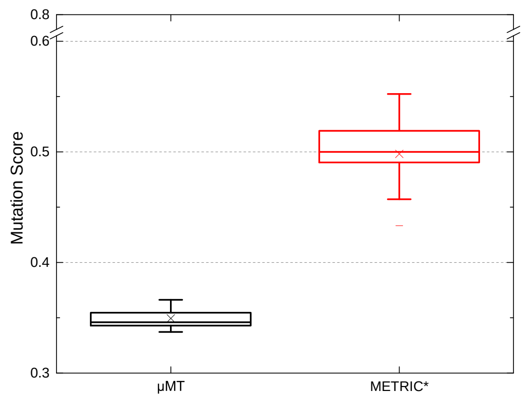


Fig. 17: Mutation score distributions of PHONE (μMT versus METRIC*)

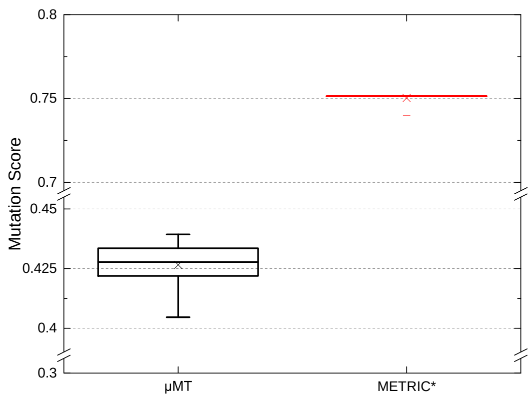


Fig. 18: Mutation score distributions of BAGGAGE (μMT versus METRIC*)

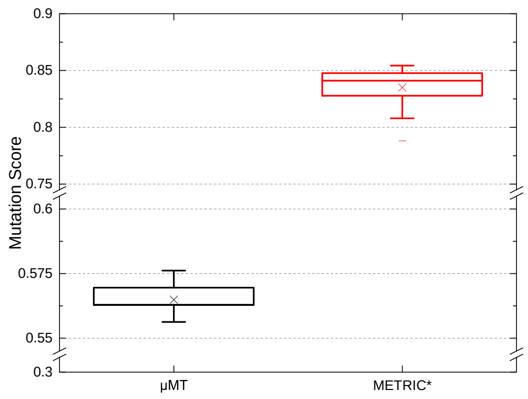


Fig. 19: Mutation score distributions of EXPENSE (μMT versus METRIC*)

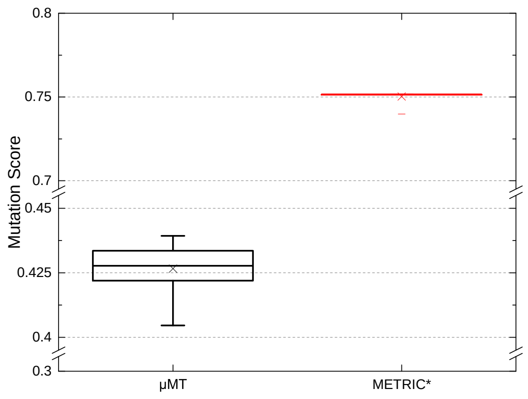


Fig. 20: Mutation score distributions of MEAL (μMT versus METRIC*)

6.6 Fault Detection Effectiveness: METRIC* versus μMT (*RQ5.2*)

We have compared the fault detection effectiveness of MRs identified by METRIC* with those by another MR identification technique, μMT. This comparison was done by computing the mutation scores of the sets of MRs (with the same size) identified by both techniques.

We used both METRIC* and μMT to identify MRs for the four subject systems. The number of MRs identified by both techniques are shown in Table 17. It was observed that, for all the subject systems, the numbers of identified MRs were much less for μMT than METRIC*. In view of this situation, for each subject system, we randomly selected the same number of MRs identified by μMT from the set of MRs generated by METRIC*. For example, for PHONE, we randomly selected 32 MRs identified by METRIC*, and used these selected MRs to compare with those 32 MRs identified by μMT. To make a fair comparison, both METRIC* and μMT generated source test cases for each subject system using the random approach. To increase the credibility of the study, for each subject system, we repeated the exercise 10 times for producing the mutation score distributions as shown in Figures 17 to 20. It was observed from these figures that, for all the four subject systems, the mutation scores (and, hence, the fault detection effectiveness) associated with METRIC* were consistently higher than those with μMT.

6.7 Threats to Validity

Below we discuss some threats of validity of our research findings.

Correctness of subject systems: With respect to the subject systems, prior to commencing our experiments, we have thoroughly checked the source code of these systems and ensured that the implementations were in line with the subject specifications.

Participant's experience gained in the study: One issue that potentially affect the validity of our study was that the participants may gain in experience on the subject specifications after doing one case. We believe that this effect should be minimal because, for each participant group, the participants were asked to identify MRs from different subject specifications between the two rounds of experiments (see Tables 8 and 9).

Coverage of various application domains: Although the four subject specifications (and their corresponding systems) were selected from different application domains, there are obviously other application domains that were not covered by these specifications. Hence, there is no complete guarantee that our study results can be generalized to all areas. However, our study results do provide an initial step towards measuring the viability and effectiveness of METRIC*.

Number of participants: Only eight participants were involved in our study. Obviously, the more participants involved in the study, the better the results. However, we would like to point out that human participants were difficult to recruit, in particular, those who are postgraduates in IT and with knowledge on software testing.

Categories and choices: The MRs identified by METRIC* and METRIC depend on the categories and choices defined from the specification. Should we allow the participants themselves to define categories and choices from the subject specifications, then our study results (in particular, the fault detection effectiveness) on the effectiveness of METRIC* would have been affected by the identified categories and choices. Consequently, this would have affected our original purpose for measuring the difference in effectiveness mainly associated with the techniques (METRIC and METRIC*) themselves. We alleviated this problem by providing the same set of IO-CTFs and complete test frames (and hence, effectively, the same set of categories and choices) to the relevant participants for each subject specification.

Assumption on the type of MRs: METRIC* assumes that an MR consists of two independent parts: a sub-relation on inputs and a sub-relation on outputs. However, some MRs cannot be split into the above two independent parts. An example is MR_2 in Example 1. This MR, $|SP(G, A, B)| = |SP(G, A, C)| + |SP(G, C, B)|$ where C is a node in $SP(G, A, B)$, has its input sub-relation and output sub-relation tangled together. To determine whether or not this type of MRs (that is, those which are difficult to be decomposed into an input sub-relation and an output sub-relation) commonly occurs, we manually analyzed 153 MRs reported in [13], [16], [17], [18], [19], [22], [23], [30], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50]. We found that only 14 MRs (about 9.2%) belong to this type. This finding indicates that METRIC* can be widely applied to many different testing situations.

Human performance data: Our study results obviously depend on the data collected and analyzed. Since our data were related to participants' (that is, human) performance, abnormal values might occasionally occur and consequently affect our analyses and conclusions. In view of this problem, in the relevant parts of our study, we repeated the exercise 10 times to reduce the impact of the abnormal data on our study results.

7 Related Work

In this section, we discuss some other MR acquisition techniques, summarize their characteristics, and distinguish them from METRIC*.

Dong et al. [50] proposed to construct new MRs through the composition of existing MRs. Two case studies were conducted and results showed that the fault detection capability of a composed MR depends on two factors: (a) the fault detection capability of those MRs which form the composed MR, and (b) the sequence of composition [50]. Liu et al. [30] further investigated the composition method for generating MRs. When the follow-up test cases of an MR can be used as the source test case of another MR, the latter relation is "composable" to the former one. The intuition is that when a series of MRs are composed to form a new one, the newly formed MR will inherit all the characteristics of the previous MRs from which the new one is generated. An experimental study was conducted and the results showed that the approach of composing MRs to form new ones can effectively improve the cost-effectiveness of MT [30]. Applying the above two composition techniques [30], [50] requires some existing MRs. METRIC* differs from them because METRIC* identifies MRs from a set of IO-CTFs and, hence, it does not need some existing MRs for its application.

Kanewala and Bieman [29] proposed an MR identification technique based on machine learning. This technique first constructs control flow graphs of functions, from which important features are extracted. These features are then used to train a predictive model using machine learning algorithms. A training set is required to support the creation of a predictive model. Once the training process is completed, a set of MRs can be identified automatically. Based on their experiments, Kanewala and Bieman [29] reported that their technique is effective in predicting MRs under certain circumstances and these predicted MRs are effective in detecting software faults. Following on their earlier work [29], Kanewala et al. [32] proposed a machine learning approach to predict MRs for scientific software based on graph kernels. Similar to the MR composition method [30], these two machine learning approaches [29], [32] require some existing MRs for their application.

Zhang et al. [31] proposed a search-based approach to automatically infer polynomial MRs for numerical input programs. This approach turns the MR inferring problem into a suitable-value searching problem by representing a particular class of MRs with a set of parameters. The program under test is executed multiple times. Execution results including inputs and outputs are then analyzed to facilitate using particle swarm optimization to solve a suitable-value searching problem. Three empirical studies were conducted and the results have shown that the inferred MRs are effective in detecting faults. In contrast to the composition methods [30], [50] and the machine learning approaches [29], [32], this search-based approach [31] does not have a prerequisite for existing MRs. Although there is no requirement for pre-existing MRs (this merit also applies to METRIC*), there is a major difference between METRIC* and this search-based approach [31]. The search-based approach requires multiple program executions for inferring a set of MRs. This requirement, however, does not exist in METRIC*.

Zhu [51] proposed a mutational MT technique to overcome the limitations of automatic test result validation in data mutation testing method. This technique uses data mutation operators, seed test cases, and mutant test cases to derive mutational MRs, which are then used to validate data mutation testing results. A test automation framework, JFuzz, was developed to support mutational MT for Java unit testing. Similarly, Sun et al. [33] proposed a data-mutation directed MR acquisition technique called μMT. This technique also uses data mutation operators to identify metamorphic relations among inputs, and uses mapping

TABLE 18: Comparison of Different MR Identification/Acquisition Techniques

MR identification/acquisition techniques	Prerequisites for application	Degree of automation
MR composition [50]	Existing MRs	Manual
MR composition [30]	Existing MRs	Manual
Machine-learning based MR detection [29]	Existing MRs	Automatic
Graph-kernel based MR prediction [32]	Existing MRs	Automatic
Search-based MR inference [31]	Multiple program executions	Automatic
Mutational MT [51]	Data mutation operators	Semi-automatic
μMT [33]	Data mutation operators and mapping rules	Semi-automatic
Output pattern-based MR identification [25]	MROPs and appropriate relations on inputs	manual
METRIC [34]	Complete test frames	Semi-automatic
METRIC*	IO-CTFs	Semi-automatic

rules of the program under test to identify the output relations of the related test cases. An empirical study involving three programs was conducted to validate the feasibility of μMT and evaluate the fault detection effectiveness of the derived MRs. The results have shown that μMT is feasible to be used for deriving MRs. The acquired MRs were found to be simple but were effective in fault detection. Although the above two techniques [33], [51] belong to the black-box approach (same as METRIC*), they are primarily based on data mutation, which is not the case for METRIC*.

Segura et al. [25] proposed an output-pattern-based MR identification approach to RESTful Web APIs. A key element of their approach is the six output patterns (MROPs) defined in terms of set relations among API responses, from which one or more specific MRs can be identified. These patterns were identified based on the observation that RESTful Web APIs have specified operations over resources (including create, read, update, and delete) and consistent parameters for standard operations (such as filtering, ordering and pagination). There is a similarity between their approach and ours: both approaches start the MR identification process with reference to the output domain, namely *output patterns* in their approach and *output test frames* in our approach. However, the approach by Segura et al. [25] focuses on the Web domain, whereas our approach (with tool support) is more generic without restricting to a particular domain as long as the category-choice framework could be applied.

Table 18 outlines some major differences among the various MR identification/acquisition techniques discussed above.

8 Summary, Conclusion, and Further Work

In this paper, we have proposed a systematic methodology, METRIC*, to help users identify MRs from specifications. An appealing characteristic of METRIC* is that it leverages the information of both inputs and outputs for MR identification. This improves the effectiveness and efficiency of the identification process. An associated generator tool MR-GEN* has been developed to automate METRIC* as far as possible.

To determine the viability and effectiveness of METRIC* (supported by MR-GEN*), an empirical study using four software systems was conducted, involving eight human participants. Our study results have demonstrated the various merits of METRIC* (for example, the high fault detection effectiveness of the generated MRs), and confirmed that the technique improves the applicability, effectiveness, and automation of MT.

In the future, we plan to investigate the various composition strategies for MRs identified by METRIC*. When the number of IO-CTFs is large, the number of identified MRs could also be large. It would be worthwhile to develop some composition strategies for these MRs to reduce the testing effort, and at the same time maintaining the fault detection effectiveness of the resulting set of MRs as far as possible. We also plan to investigate MR prioritization techniques, which are highly desired when testing resources are tight. Finally, for the 153 MRs we manually identified from the literature review, we plan to further analyze and classify them into different types. This will help us identify potentially useful MR patterns (or templates) to facilitate subsequent MR identification.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grant No. 61872039), the Beijing Natural Science Foundation of China (Grant No. 4162040), the Aeronautical Science Foundation of China (Grant No. 2016ZD74004), the Fundamental Research Funds for the Central Universities (Grant No. FRF-GF-17-B29).

References

[1] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing without the need of oracles," *Information and Software Technology*, vol. 45, no. 1, pp. 1–9, 2003.
[2] K. Patel and R. M. Hierons, "A mapping study on testing non-testable systems," *Software Quality Journal*, 2017. [Online]. Available: https://doi.org/10.1007/s11219-017-9392-4
[3] D. S. Rosenblum, "A practical approach to programming with assertions," *IEEE Transactions on Software Engineering*, vol. 21, no. 1, pp. 19–31, 1995.
[4] L. Manolache and D. G. Kourie, "Software testing using model programs," *Software: Practice and Experience*, vol. 31, no. 13, pp. 1211–1236, 2001.
[5] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.
[6] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: a review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1–4:27, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3143561
[7] P. Hu, Z. Zhang, W. K. Chan, and T. H. Tse, "An empirical comparison between direct and indirect test result checking approaches," in *Proceedings of the 3rd International Workshop on Software Quality Assurance (SOQUA 2006)*. ACM, 2006, pp. 6–13.
[8] J. C. Knight and N. G. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," *IEEE Transactions on Software Engineering*, vol. 12, no. 1, pp. 96–109, 1986.

[9] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, "How effectively does metamorphic testing alleviate the oracle problem?" *IEEE Transactions on Software Engineering*, vol. 40, no. 1, pp. 4–22, 2014.

[10] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[11] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering (ICSE 2018)*. ACM, 2018, pp. 303–314.

[12] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Communications of the ACM*, in press.

[13] V. Le, M. Afshari, and Z. Su, "Compiler validation via equivalence modulo inputs," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2014)*, vol. 49, no. 6. ACM, 2014, pp. 216–226.

[14] A. F. Donaldson and A. Lascu, "Metamorphic testing for (graphics) compilers," in *Proceedings of the 1st International Workshop on Metamorphic Testing (MET 2016), in conjunction with the 38th International Conference on Software Engineering (ICSE 2016)*. ACM, 2016, pp. 44–47.

[15] A. F. Donaldson, H. Evrard, A. Lascu, and P. Thomson, "Automated testing of graphics shader compilers," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 93:1–93:29, 2017.

[16] T. Y. Chen, J. W. Ho, H. Liu, and X. Xie, "An innovative approach for testing bioinformatics programs using metamorphic testing," *BMC Bioinformatics*, vol. 10, no. 1, p. 24, 2009.

[17] L. L. Pullum and O. Ozmen, "Early results from metamorphic testing of epidemiological models," in *Proceedings of the 2012 ASE/IEEE International Conference On BioMedical Computing (BioMedCom 2012)*. IEEE, 2012, pp. 62–67.

[18] W. K. Chan, T. Y. Chen, H. Lu, T. Tse, and S. S. Yau, "Integration testing of context-sensitive middleware-based applications: a metamorphic approach," *International Journal of Software Engineering and Knowledge Engineering*, vol. 16, no. 5, pp. 677–703, 2006.

[19] M. Jiang, T. Y. Chen, F.-C. Kuo, and Z. Ding, "Testing central processing unit scheduling algorithms using metamorphic testing," in *Proceedings of the 4th IEEE International Conference on Software Engineering and Service Science (ICSESS 2013)*. IEEE, 2013, pp. 530–536.

[20] T. Y. Chen, J. Feng, and T. H. Tse, "Metamorphic testing of programs on partial differential equations: a case study," in *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC 2002)*. IEEE, 2002, pp. 327–333.

[21] C.-A. Sun, Z. Wang, and G. Wang, "A property-based testing framework for encryption programs," *Frontiers of Computer Science*, vol. 8, no. 3, pp. 478–489, 2014.

[22] T. Y. Chen, F.-C. Kuo, W. Ma, W. Susilo, D. Towey, J. Voas, and Z. Q. Zhou, "Metamorphic testing for cybersecurity," *Computer*, vol. 49, no. 6, pp. 48–55, 2016.

[23] W. K. Chan, S. C. Cheung, and K. R. Leung, "A metamorphic testing approach for online testing of service-oriented software applications," *International Journal of Web Services Research*, vol. 4, no. 2, pp. 61–81, 2007.

[24] C.-A. Sun, G. Wang, B. Mu, H. Liu, Z. S. Wang, and T. Y. Chen, "A metamorphic relation-based approach to testing web services without oracles," *International Journal of Web Services Research*, vol. 9, no. 1, pp. 51–73, 2012.

[25] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Corts, "Metamorphic Testing of RESTful Web APIs," *IEEE Transactions on Software Engineering*, 2017. [Online]. Available: https://doi.org/10.1109/TSE.2017.2764464

[26] "GraphicsFuzz is acquired by Google," last accessed on August 19, 2018. [Online]. Available: https://www.graphicsfuzz.com/

[27] "How it works," last accessed on August 29, 2018. [Online]. Available: https://www.graphicsfuzz.com/howitworks.html

[28] A. Groce, T. Kulesza, C. Zhang, S. Shamasunder, M. Burnett, W.-K. Wong, S. Stumpf, S. Das, A. Shinsel, F. Bice *et al.*, "You are the only possible oracle: effective test selection for end users of interactive machine learning systems," *IEEE Transactions on Software Engineering*, vol. 40, no. 3, pp. 307–323, 2014.

[29] U. Kanewala and J. M. Bieman, "Using machine learning techniques to detect metamorphic relations for programs without test oracles," in *Proceedings of the 24th International Symposium on Software Reliability Engineering (ISSRE 2013)*. IEEE, 2013, pp. 1–10.

[30] H. Liu, X. Liu, and T. Y. Chen, "A new method for constructing metamorphic relations," in *Proceedings of the 12th International Conference on Quality Software (QSIC 2012)*. IEEE, 2012, pp. 59–68.

[31] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei, "Search-based inference of polynomial metamorphic relations," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE 2014)*. ACM,2014, pp. 701–712.

[32] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels," *Software Testing, Verification and Reliability*, vol. 26, no. 3, pp. 245–269, 2016.

[33] C.-A. Sun, Y. Liu, Z. Wang, and W. Chan, "μMT: a data mutation directed metamorphic relation acquisition methodology," in *Proceedings of the 1st International Workshop on Metamorphic Testing, in conjunction with the 38th International Conference on Software Engineering (ICSE 2016)*. ACM, 2016, pp. 12–18.

[34] T. Y. Chen, P.-L. Poon, and X. Xie, "METRIC: METamorphic Relation Identification based on the Category-choice framework," *Journal of Systems and Software*, vol. 116, pp. 177–190, 2016.

[35] T. Y. Chen, P.-L. Poon, and T. H. Tse, "A choice relation framework for supporting category-partition test case generation," *IEEE Transactions on Software Engineering*, vol. 29, no. 7, pp. 577–593, 2003.

[36] P.-L. Poon, S.-F. Tang, T. H. Tse, and T. Y. Chen, "CHOC'LATE: a framework for specification-based testing," *Communications of the ACM*, vol. 53, no. 4, pp. 113–118, 2010.

[37] H. Liu, P.-L. Poon, and T. Y. Chen, "Enhancing partition testing through output variation," in *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*. IEEE, 2015, pp. 805–806.

[38] Y.-S. Ma, J. Offutt, and Y.-R. Kwon, "muJava: a mutation system for java," in *Proceedings of the 28th International Conference on Software Engineering (ICSE 2006)*. ACM, 2006, pp. 827–830.

[39] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Semi-proving: an integrated method based on global symbolic evaluation and metamorphic testing," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 4, pp. 191–195, 2002.

[40] M. Jiang, T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Ding, "A metamorphic testing approach for supporting program repair without the need for a test oracle," *Journal of systems and software*, vol. 126, pp. 127–140, 2017.

[41] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Semi-proving: an integrated method for program proving, testing, and debugging," *IEEE Transactions on Software Engineering*, vol. 37, no. 1, pp. 109–125, 2011.

[42] Z. Q. Zhou, S. Zhang, M. Hagenbuchner, T. H. Tse, F.-C. Kuo, and T. Y. Chen, "Automated functional testing of online search services," *Software Testing, Verification and Reliability*, vol. 22, no. 4, pp. 221–243, 2012.

[43] X. L. Lu, Y. W. Dong, and C. Luo, "Testing of component-based software: a metamorphic testing methodology," in *Proceedings of the 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC 2010)*. IEEE, 2010, pp. 272–276.

[44] X. Xie, W. E. Wong, T. Y. Chen, and B. Xu, "Spectrum-based fault localization: testing oracles are no longer mandatory," in *Proceedings of the 11th International Conference on Quality Software (QSIC 2011)*. IEEE, 2011, pp. 1–10.

[45] F.-C. Kuo, T. Y. Chen, and W. K. Tam, "Testing embedded software by metamorphic testing: a wireless metering system case study," in *Proceedings of the 36th Conference on Local Computer Networks (LCN 2011)*. IEEE, 2011, pp. 291–294.

[46] S. Segura, R. M. Hierons, D. Benavides, and A. Ruiz-Cortés, "Automated metamorphic testing on the analyses of feature models," *Information and Software Technology*, vol. 53, no. 3, pp. 245–258, 2011.

[47] X. Xie, W. E. Wong, T. Y. Chen, and B. Xu, "Metamorphic slice: an application in spectrum-based fault localization," *Information and Software Technology*, vol. 55, no. 5, pp. 866–879, 2013.

[48] H. Liu, I. I. Yusuf, H. W. Schmidt, and T. Y. Chen, "Metamorphic fault tolerance: an automated and systematic methodology for fault tolerance in the absence of test oracle," in *Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion 2014)*. ACM, 2014, pp. 420–423.

[49] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic

testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

[50] G.-W. Dong, B.-W. Xu, L. Chen, C.-H. Nie, and L.-L. Wang, "Case studies on testing with compositional metamorphic relations," *Journal of Southeast University (English Edition)*, vol. 24, no. 4, pp. 437–443, 2008.

[51] H. Zhu, "JFuzz: a tool for automated java unit testing based on data mutation and metamorphic testing methods," in *Proceedings of the 2nd International Conference on Trustworthy Systems and their Applications (TSA 2015)*. IEEE, 2015, pp. 8–15.

Chang-ai Sun is a Professor in the School of Computer and Communication Engineering, University of Science and Technology Beijing. Before that, he was an Assistant Professor at Beijing Jiaotong University, China, a postdoctoral fellow at the Swinburne University of Technology, Australia, and a postdoctoral fellow at the University of Groningen, The Netherlands. He received the bachelor degree in Computer Science from the University of Science and Technology Beijing, China, and the PhD degree in Computer Science from the Beihang University, China. His research interests include software testing, program analysis, and Service-Oriented Computing.

An Fu is a PhD student in the School of Computer and Communication Engineering, University of Science and Technology Beijing, China. He received the bachelor degree in Information Security from University of Science and Technology Beijing, China. His current research interests include software testing and debugging.

Pak-Lok Poon received his PhD degree in Software Engineering from The University of Melbourne. He is currently an associate professor in the School of Engineering and Technology at CQUniversity Australia. His research interests include software testing, requirements engineering and inspection, information systems audit and control, electronic commerce, and computers in education. He was a guest editor of the special issue of the *Journal of Systems and Software* on test oracles in 2018. He served on the editorial board of the *Information Systems Control Journal*, and is currently an editorial review board member of the *Journal of Organizational and End User Computing* and the *International Journal of Information Systems and Supply Chain Management*. He is a member of the IEEE and the ACM.

Xiaoyuan Xie received B.Sc. and M.Phil. degrees in Computer Science from Southeast University, China in 2005 and 2007, respectively, and received PhD degree in Computer Science from Swinburne University of Technology, Australia in 2012. She is currently a professor in School of Computer Science, Wuhan University, China. Her research interests include software analysis, testing, debugging, and search-based software engineering.

Huai Liu is a Lecturer of Information Technology at the College of Engineering & Science in Victoria University, Melbourne, Australia. Prior to joining VU, he worked as a research fellow at RMIT University and a research associate at Swinburne University of Technology. He received the BEng in physioelectronic technology and MEng in communications and information systems, both from Nankai University, China, and the PhD degree in software engineering from the Swinburne University of Technology, Australia. His current research interests include software testing, cloud computing, and end-user software engineering.

Tsong Yueh Chen is a Professor of Software Engineering at the Department of Computer Science and Software Engineering in Swinburne University of Technology. He received his PhD in Computer Science from The University of Melbourne, the MSc and DIC from Imperial College of Science and Technology, and BSc and MPhil from The University of Hong Kong. He is the inventor of adaptive random testing and metamorphic testing. His main research interest is software testing.