Metamorphic Relations for Enhancing System Understanding and Use

Zhi Quan Zhou, Liqun Sun, Tsong Yueh Chen, and Dave Towey

Abstract—Modern information technology paradigms, such as online services and off-the-shelf products, often involve a wide variety of users with different or even conflicting objectives. Every software output may satisfy some users, but may also fail to satisfy others. Furthermore, users often do not know the internal working mechanisms of the systems. This situation is quite different from bespoke software, where developers and users usually know each other. This paper proposes an approach to help users to better understand the software that they use, and thereby more easily achieve their objectives—even when they do not fully understand how the system is implemented. Our approach borrows the concept of metamorphic relations from the field of metamorphic testing (MT), using it in an innovative way that extends beyond MT. We also propose a "symmetry" metamorphic relation pattern and a "change direction" metamorphic relation input pattern that can be used to derive multiple concrete metamorphic relations. Empirical studies reveal previously unknown failures in some of the most popular applications in the world, and show how our approach can help users to better understand and better use the systems. The empirical results provide strong evidence of the simplicity, applicability, and effectiveness of our methodology.

Index Terms—Metamorphic exploration, symmetry, metamorphic testing, metamorphic relation, metamorphic relation pattern, metamorphic relation input pattern, change direction, oracle problem, user experience, user countermeasure, software validation.

◆

1 INTRODUCTION

It is generally agreed that the main challenge for software and systems engineering is not the limited raw computing resources, but rather our limited ability to correctly construct complex systems. In the context of software engineering, an *oracle* is a mechanism against which the correctness of the software behavior can be decided. Having a test oracle (automated, if possible) is essential to successful testing. In many settings, however, such an oracle is not available, or is theoretically available, but practically too difficult or expensive to be applied. This is a situation known as the *oracle problem*, which is a major challenge in software testing.

There is a growing interest in test oracles from the research community, as reflected by an increasing number of research activities involving this topic in recent years [1]–[8]. Most of these research activities address the oracle problem from the developers' or testers' perspective for the purpose of software testing and verification. Barr et al. [4], for instance, identified four broad categories of oracles: specified oracles (such as assertions and algebraic specifications); derived oracles (such as metamorphic relations in metamorphic testing, and invariants in invariant detection); implicit oracles (such as the detection of crashes by fuzzing); and human oracles. Each type of oracle has its own advantages and limitations, as well as open research questions.

In contrast to the existing literature, in this research we go beyond conventional software verification and validation to consider the oracle problem from a different perspective, and for a different purpose: Taking the *users'* perspective, we aim to explore the software system with the goal of helping users to better understand it, and thereby make better use of it for their own specific information needs.

Our motivation for this research is explained as follows: In traditional development of bespoke systems, software is produced by a specific developer (such as a software house or an information technology (IT) department) for a specific organization or user. In such a paradigm, the developer and user may know each other quite well. This is different from the situation where software is developed for the mass market (such as commercial off-the-shelf packages, open-source software, and free software), where there is a diverse user base. Furthermore, in modern IT paradigms such as online services and cloud computing, an application often has many different users using it for different, or even conflicting, purposes. On the one hand, the developer may not necessarily understand all of the users' needs and objectives (for example, what a user is really looking for when shopping online); on the other hand, as pointed out by Zhou et al. [9], the users often lack a comprehensive system specification (apart from an often brief user manual or online help page) that can help them to understand how the system really works (for example, how an eCommerce website ranks the products). This all means that, while

Zhi Quan Zhou and Liqun Sun are with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia. E-mails: zhiquan@uow.edu.au, ls168@uowmail.edu.au.

Tsong Yueh Chen is with the Department of Computer Science and Software Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia. E-mail: tychen@swin.edu.au.

Dave Towey is with the School of Computer Science, University of Nottingham Ningbo China, Ningbo, Zhejiang 315100, China. E-mail: Dave.Towey@nottingham.edu.cn.

the produced software may satisfy some users, it may also (simultaneously) fail to satisfy others. Furthermore, as will be shown in this paper, even the most popular software systems may contain serious faults. Due to the oracle problem, however, it may not be possible for the user to identify when an output is erroneous. As such, we argue that users need to operate software in a smarter way, to both best satisfy their own needs, and to enable awareness of erroneous software behavior and output.

It should be noted that software for the mass market has been developed for several decades, and understanding the needs of a diverse user base has long been recognized as a challenge. Lehman [10] classified computer programs according to their relationship to the environment in which they are executed, and defined *E-programs* to be those that mechanize a human or societal activity (such as operating systems, air-traffic control, and stock control). Lehman pointed out that the support required of such programs would depend on program characteristics as experienced by the users. He further pointed out that the validity of E-programs depends on human assessment of its effectiveness in the intended application, and that the "proof of correctness" of such programs is basically irrelevant—a "correct" program may be useless, and an "incorrect" program may be quite usable: "it is the detailed *behavior* of the program *under operational conditions* that is of concern" [10, p. 1064]. Lehman's work strongly contributed to the identification of sources of evolutionary pressure on computer software, and explained why software maintenance is a never-ending process.

It should be pointed out, however, that while developers strive to satisfy, as much as possible, the users' requirements during software evolution, the complexity (and quantity) of users' needs makes it difficult to target each and every user in the mass market. Therefore, from a specific user's perspective, there is a need for a practical method that can help to give insight into the software. The perspective of the present research, therefore, is different from that of Lehman [10]: Instead of trying to help the developers to understand the users' needs, we attempt to help the users to understand the software, and hence make better use of it. This means that our techniques will also be useful for the developers to train their clients and testers, such as in the context of user acceptance testing [11], alpha/beta testing, and crowdsourced testing [12], [13]. Section 10.4 presents further discussion of users' roles.

In the area of software reverse engineering, intensive research has been conducted into program understanding / comprehension [14]–[17], which is a key activity in software maintenance: Software must be sufficiently understood before it can be properly modified [16]. The objective of the present research differs from that of the program understanding literature because our aim is to help the users to better understand and use the system, rather than help the developers to comprehend the source code or programming interfaces for software

maintenance purposes. Nevertheless, this research also benefits software comprehension as we develop a black-box approach for understanding a system in the absence of a comprehensive specification.

Our research questions are stated as follows:

- **RQ1**: Can we design a mechanism to help users quickly recognize whether the software, or the software output, is really appropriate for their needs?
- **RQ2**: Can our mechanism help users to obtain more desirable computation results, even if the software being used is defective, or the user does not fully understand how it actually works?
- **RQ3**: Because we target end users, our solutions must be simple, applicable, and effective, for multiple scenarios and application domains. To what degree do our solutions possess these properties?

The contributions of this research are summarized as follows:

- We propose an innovative way of exploring software systems, called *metamorphic exploration*, that can help users to better understand the systems and better achieve their objectives, even if the software is not well documented. Our approach borrows the concept of *metamorphic relations* (MRs) from the field of *metamorphic testing* (MT), using the MRs for purposes beyond conventional software testing and verification.
- We further propose the concepts of a "symmetry"[1] *metamorphic relation pattern* (MRP), and of a "change direction" *metamorphic relation input pattern* (MRIP). MRPs and MRIPs can be used either separately or in combination to derive various concrete metamorphic relations across many different application domains.
- We discuss case studies conducted using a variety of very popular software applications. These applications are quite different in nature, and include 65 of the top commercial websites, Google Maps navigation, Google Maps location-based search, image analysis software for face recognition (including MATLAB, OpenCV, and Facebook), and the Google video analysis service Cloud Video Intelligence. We show how "symmetry" MRs can help users to (1) detect previously unknown failures in each of these applications, and (2) obtain more desirable computation results in spite of the failures, even when the users do not fully understand how the software is actually implemented.
- The empirical results provide strong evidence of the simplicity, wide applicability, and effectiveness of

---

1. In this paper, we use the word "symmetry" to refer to both an intrinsic property of the real world (including nature, human behavior, and human society), and a desirable property of computer systems. Our concept of "symmetry," therefore, has a much broader meaning than that used in previous studies in the software testing and verification literature, such as Gotlieb's [18] (where "symmetries" referred to the permutation relation of program input and output vectors) and that of Ip and Dill [19] (where, based on results from group theory, structural symmetries were exploited to address the state explosion problem in model checking).

our methodology, hence providing an affirmative answer to all three research questions.

The evaluation and limitations of our approach are summarized as follows (with further details given in Section 11): On the one hand, feedback and news from the real world confirm the validity and usefulness of our results; on the other hand, it should be noted that our approach is only a partial analysis technique, whose effectiveness is dependent on the chosen MRs, the chosen inputs, as well as the background, experience, and perspectives of the users—as will be discussed, however, the MR and input choice, and even the users' initial familiarity with the concepts, are issues easily addressed, often within only a few hours training [20]–[22]. The scale of our case studies is also limited and, therefore, there is a threat to the external validity of our results, in terms of the generalization of the findings to other areas or other software features not yet investigated.

The rest of the paper is organized as follows: Section 2 reviews the basic concepts of MT and MRs. Section 3 introduces our approach, using motivating, real-life examples. Section 4 discusses our MR patterns, the symmetry MRP and the "change direction" MRIP. Sections 5 to 9 present a series of case studies using popular applications from various domains. Section 10 contains further discussion of related topics, including the relationships between the present work and metamorphic testing, the validity and sufficiency of MRs, the design space and choices, and customers' roles in related software processes. Section 11 presents evaluation and limitations of our approach. Finally, Section 12 concludes the paper.

2 METAMORPHIC TESTING (MT)

Because our research borrows from some of the core concepts in the field of MT, we therefore first give a brief review of some of the principles of MT.

MT is a property-based software testing method [23], [24]. It differs from conventional testing techniques in that it does not focus on the verification of each individual output of the software under test (SUT), but instead checks the *relations* among the inputs and outputs of *multiple* executions of the SUT. Such relations are called *metamorphic relations* (MRs). MRs are necessary properties of the intended program's functionality. If an MR is violated for certain test cases, then the SUT must be at fault.

Consider, for example, a program $p(G,\ a,\ b)$ that identifies the shortest path from an origin node a to a destination node b in an undirected graph G. When G is large and complex, and when a and b are chosen randomly, it is hard to verify the output of p because of the lack of a practical oracle. Nevertheless, we can identify some MRs for the shortest path problem. One of the MRs can state, for example, that swapping the origin and destination nodes should not affect the length of the shortest path [25]. Based on this MR, MT can be performed by running the program p twice: once for a

source execution, on a *source input* $(G,\ a,\ b)$ to produce a *source output*; and once for a *follow-up execution*, on a *follow-up input* $(G,\ b,\ a)$ to produce a *follow-up output*. If a source output and its follow-up output are found to have different lengths, then we say that the MR has been violated and, hence, a fault in p has been revealed. Many different MRs can be identified for the shortest path problem [25]. Interested readers are referred to Chen et al. [26] for formal definitions of the concepts. Section 10.2 answers further questions relating to MRs.

A growing body of research has examined the concept of MT [6], [26]–[29], and proven it to be a very useful testing paradigm that effectively addresses the oracle problem. The increasing interest in MT is not only because of its ability to test software in the absence of an ideal oracle, but also because MT is based on a perspective not previously used by other testing techniques—as a result, it has been reported that MT can detect previously unknown faults in mature software systems such as the Siemens suite of programs [30], the GCC and LLVM compilers [3], [31], graphics shader compilers [32], various open source and commercial code obfuscators [33], and the Google Maps navigation system [34].

When studying MT for machine learning software, Xie et al. [35] found that an MR violation could reveal problems not only in the implementation but also in the choice of algorithm. In this situation, the MR was identified based on the expected functionality of the system: A violation of this MR, therefore, could indicate that the chosen algorithm was not appropriate (and not just that the implementation had faults). Xie et al.'s study was at the algorithm selection level, checking whether or not the adopted algorithm was appropriate. More recently, Zhou et al. [9] expanded MT into a unified framework that covers verification, validation, and other types of software quality assessment. They showed that MRs could be identified by users based on their expectations (reflecting what they really cared about), rather than being based on the system designs chosen by the system designer or developer. Their study used MT at the top (system/services) level, and involved very large scale empirical studies with major web search engines. It should be noted that these studies [9], [35] did not address how to obtain better computation results from the users' perspective, nor any of the research questions raised in Section 1.

3 OUR APPROACH

In the following, we use two real-life examples to illustrate the basic ideas of our approach.

3.1 Example 1: Searching for American Inventors

Suppose that a student wants to find a list of the top American inventors of all time. She goes to www.google.com and enters the following query:

[`American inventors`].[2] Google immediately returns a list of top American inventors under the category "Inventors > United States" as shown in Fig. 1a. Suppose the student wants to check whether these Google results are stable, or whether Google can provide a different list. She therefore identifies an MR that she thinks should hold: Searching in a different language should return the same, or a similar, list of American inventors. Using Google Translate, the student then translates [`American inventors`] into simplified Chinese: [美国发明家]. She copies and pastes these Chinese characters into Google, and searches again. To her surprise, this time Google returns a very different list of American inventors, still under the category "Inventors > United States," as shown in Fig. 1b.[3]

The student cannot understand why there is such a dramatic difference between the two lists provided by Google. Why, for example, is Thomas Edison listed as the #1 American inventor for the Chinese query, but he does not even appear among the top five for the English query? Nevertheless, the student has now obtained a deeper understanding of how Google works: It gives different lists for different query languages, even though the queries appear equivalent, and the categorizations ("Inventors > United States") are also identical. The student can now decide whether she still wants to use the Google results (for example, by choosing one of the two lists, or by combining them), or can choose to consult a different website.

3.2 Example 2: Machine Translation

Suppose a user wants to translate the Chinese sentence 我爱吃苹果 into English. Because he does not know any English, he decides to use a machine translation service, and so goes to www.bing.com, and enters the following query: [`translate` 我爱吃苹果]. As shown in Fig. 2a, Bing translates the sentence into English as: `I love to eat.`

Using an MR to check this result, the user translates the English sentence `I love to eat` into Chinese to see whether the resulting Chinese sentence has a similar meaning to the original one. To the user's surprise, the resulting Chinese sentence is: 我爱吃, as shown in Fig. 2b. He notes that the last two Chinese characters in his original sentence, 苹果 (apple), are missing, and hence has found an error in the translation results. Although he cannot tell whether the error was in the Chinese-English or English-Chinese translation, he decides to use other machine translation services for which the MR holds.[4]

---

2. A note on the square brackets convention: Although a query can be surrounded by square brackets in a specification, the brackets should not be typed when doing the actual query.

3. A similar observation has been reported by Sailer [36]. The inconsistencies can also be produced using other queries, such as [`British Engineers`].

4. The correct translation should be "I love to eat apples." We found that this translation error remained in Bing for more than a week in October 2017, caused by a fault in the Microsoft Translator API (https://www.microsoft.com/en-us/translator/translatorapi.aspx).
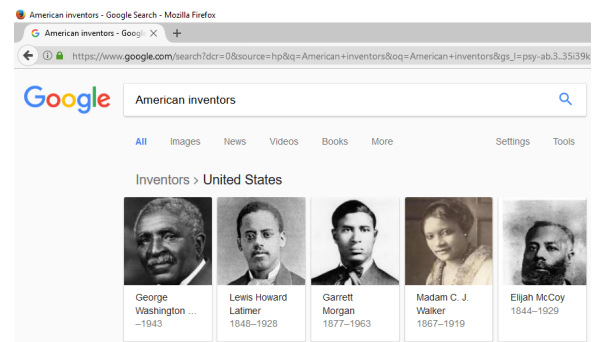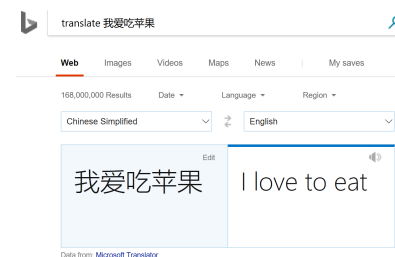

(a) Top American inventors returned by Google, under the category "Inventors > United States."
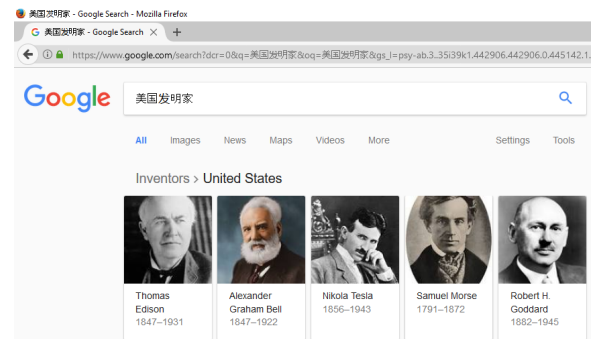
(b) An equivalent query term in Chinese yielded a very different list of American inventors, still under the category "Inventors > United States."

Fig. 1: A metamorphic relation helped the user to expand her Google search results.

(a) Using Bing to translate a Chinese sentence into English.
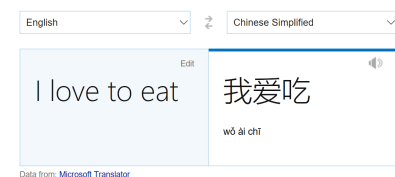
(b) Translating the English sentence back to Chinese revealed an error.

Fig. 2: A metamorphic relation helped the user to identify an error in Bing translation.

4 METAMORPHIC RELATION PATTERN (MRP)

A challenge in our approach, as well as in MT in general, is the systematic identification of effective MRs. Abstract MRs, which can be used to derive multiple concrete MRs, appear to be a possible solution for this. In Section 4.1, we review the background of this research direction. Then, in Section 4.2, we reflect on two independent studies where MT was successfully applied to detect real-life bugs in very different application domains. We provide our insights, and discuss a discovered pattern underlying both of these studies. We call this pattern *symmetry*, and define it in Section 4.3. In Section 4.4, we show that symmetry is indeed an intrinsic property of the real world (evident in places including nature and human society). Some of the concepts or principles presented in this section are used later in this paper in the context of software systems. In Section 4.5, we propose using symmetry as a fundamental *metamorphic relation pattern* (MRP), motivated by the insight that the virtual world (computer systems) is built to simulate, model, meet user requirements, or solve problems from the real world. Finally, in Section 4.6, we propose *metamorphic relation input pattern* (MRIP).

4.1 Background

In early MT research, MRs were usually identified from scratch for each individual problem under study. To assist with systematic identification of MRs, Zhou et al. were the first to propose a concept of *general metamorphic relations* [37, p. 3], which is an abstract form of MR that can be used to derive multiple concrete MRs. For the purpose of testing search engines, they described a "general metamorphic relation" as follows:

> MR_{SEARCH}: If search criterion X_2 implies search criterion X_1 (that is, if X_2 is satisfied then X_1 is satisfied), then $\#(X_2) \leq \#(X_1)$, where $\#(X)$ denotes the number of results satisfying the search criterion X.

Using this general MR, many concrete MRs can be derived [37], such as:

MR_{OR}: If $A_1 \equiv (A_2 \text{ OR } B)$, then $\#(A_2) \leq \#(A_1)$, where A_1, A_2, and B denote search criteria / conditions, and "OR" is the logical operator for inclusive disjunction ($\vee$). Similarly, we can have the following concrete MR:

MR_{AND}: If $A_1 \equiv (A_2 \text{ AND } B)$, then $\#(A_1) \leq \#(A_2)$, where "AND" is the operator for logical conjunction ($\wedge$).

In a follow-up study, Zhou et al. further identified a pattern (instead of using the word "pattern," they called it a "general relation" [38, p. 223]), which is a *subset* relation:

$$Pages(X_2) \subseteq Pages(X_1),$$

where $Pages(X_1)$ denotes the source output (that is, search results, or web pages in the context of a web search engine, satisfying the search criterion X_1) and $Pages(X_2)$ denotes the follow-up output (search results satisfying the search criterion X_2).

Empirical results showed that concrete MRs derived from the above abstract forms of MRs had a strong fault-detection capability [37], [38]. For example, Microsoft's Live Search returned $11,783$ results for the query [GLIF] but zero results for [GLIF OR 5Y4W] (where "OR" is a reserved keyword of the search engine for inclusive disjunction)—this obviously violated MR_{OR} and, because the violation was repeatable, a fault in Live Search was revealed [38].

A limitation of Zhou et al.'s work [37], [38] is that they only considered a specific type of software under test (namely, software performing *search* operations, of which each output was a list of search results) and, therefore, their patterns were not general enough to cover other types of functions, features, and application domains.

Also using abstract MRs to derive concrete ones for systematic MR generation, Segura et al. [39] explicitly proposed *metamorphic relation output pattern* (MROP) in the context of testing RESTful web APIs (that implement create, read, update, or delete operations over a resource). They stated that an MROP "defines an abstract output relation typically identified in Web APIs" and that "each MROP is defined in terms of set operations among test outputs such as equality, union, subset, or intersection." They found MROPs to be very helpful for deriving concrete MRs. More specifically, they identified six MROPs: (1) Equivalence (representing relations where the source and follow-up outputs include the same items although not necessarily in the same order); (2) Equality (representing relations where the source and follow-up outputs must contain the same items, in the same order); (3) Subset (representing subset relations among the source and follow-up outputs, similar to the "general relation" ($Pages(X_2) \subseteq Pages(X_1)$) proposed by Zhou et al. [38, p. 223]); (4) Disjoint (representing relations where the source and follow-up outputs should be disjoint sets—having no elements in common); (5) Complete (representing relations where the union of the follow-up outputs should contain the same items as the source output); and (6) Difference (representing relations where the source and follow-up outputs should differ in a specific set of items).

A concrete example of an "equivalence" MROP is that a search for YouTube videos should return the same set of videos "regardless of the ordering criteria specified (date, rating, relevance, title, or number of views)" [39]. Segura et al. [39] hypothesized that their proposed patterns could also be useful for automated inferencing of likely MRs for a given API; however, they also pointed out that such research, as in [40]–[42], was challenging, and still at an early stage.

Compared with Zhou et al.'s initial work on this topic [37], [38], Segura et al. [39] investigated the pattern concept more explicitly and systematically, providing strong evidence of its potential usefulness. However, Segura et al.'s patterns face a similar limitation to that of Zhou et al.: They were only designed for a specific type of program (RESTful web APIs, performing one of four

operations over a resource—create, read, update, and delete) with an emphasis on the search (read) operation [39]. Furthermore, Segura et al. [39] only studied patterns for output relations, and the outputs must be sets, and the output relations must be defined in terms of set operations. These limitations mean that all previous work [37]–[39] has a significant applicability problem and, therefore, cannot be used to address our research questions RQ1, RQ2, and RQ3, which require a solution that should not put any restriction on the type of application, feature, function, operation, input/output format, or problem domain. In this paper, we propose generic patterns (including one MRP and one MRIP) that have no such restrictions.

4.2 Reflection on Two Independent Studies

In 2017, two independent MT studies [6], [43] were presented at the ICSE metamorphic testing workshop that showed great research serendipity: They both employed geometric transformations to transform the source input into the follow-up input, and very effectively detected software faults. In Lindvall et al.'s work [43], researchers from the Fraunhofer Center for Experimental Software Engineering tested autonomous drones using MRs that "leverage geometric changes in the simulated environment, such as rotation and translation in combination with different formations of obstacles in the scenarios that the drone exposed to." For example, for a given source input (scenario), a follow-up input (scenario equivalent to the source one) could be generated by rotating the world geometry of the source scenario while keeping the distances and relative positions unchanged. The drone was expected to behave consistently under the source and follow-up scenarios. For instance, other conditions being the same, the drone should have similar behavior regardless of whether it was flying north or south. Using this kind of MR, the researchers detected critical faults that could cause crashes: "instead of avoiding the obstacle, the drone flew straight into it and crashed."

Independent of Lindvall et al.'s work, engineers from Adobe [6] used MT to test the *time series analysis* (TSA) service of Adobe Marketing Cloud (http://www.adobe.com/marketing-cloud.html), software that provides customers with automatic identification and reporting of anomalies in marketing data. Input for a TSA request includes, among others, training data and metric data (data to be modeled). The training data are a set of values at different time points, used to train the statistical model. Once a model is selected as the best fit for the training data, it is then applied to the metric data to predict future behavior.

To test the TSA, Jarman et al. [6] treated every time series (input to the SUT) as a 2D geometric object, assigning the x-axis as time t and the y-axis as the respective data value. They pointed out that "if the assumption is made that all statistical models are derived from the internal relationships of the values . . . we can safely expect that TSA will produce identical models regardless of orientation in space." As such, they defined their MR as: Given a time series $f(x)$ (source input) and its expected TSA model $m(f(x))$ (source output), applying a geometric transformation T to $f(x)$, denoted by $T(f(x))$ (follow-up input), should have an expected TSA model (follow-up output) equal to $T(m(f(x)))$, that is: $T(m(f(x))) = m(T(f(x)))$. Using this MR, together with a set of very simple source test cases, the engineers detected three previously unknown bugs in the TSA software and reported that the MR "has been proven highly effective as shown by the high violation rates."

It should be noted that these two studies [6], [43] were conducted independently, by two different teams, in completely different application domains. However, both studies showed that MRs based on geometric transformations were very effective for fault detection: In Lindvall et al.'s work [43], the follow-up input (scenario) was generated by applying geometric transformations to the source input (scenario); whereas in Jarman et al.'s study [6], the source input was $f(x)$ (the 2D geometric object representing a time series), and the follow-up input, $T(f(x))$, was generated by applying the geometric transformation T to the source input. The MRs used in both these two studies shared a similar viewpoint: The system should appear more or less the same under geometric transformations.

Are the observations of the effectiveness of geometric transformations just a coincidence, happening by chance in both of these different application areas? A reflection on this question led us to the conclusion that geometric transformations are an instance of a more fundamental and pervasive property of the real world: *Symmetry*. The discovery of the power of geometric transformations in MT, therefore, might have revealed part of the usefulness of symmetry as a generic MR pattern. In fact, we found that the geometric transformations and the examples of Sections 3.1 and 3.2 are all instances of symmetry. Before we go deeper into this topic, we wish to first review the basic concept of symmetry.

4.3 Definition of Symmetry

Symmetry is an immensely important concept in the sciences and the arts, and can be defined in different ways, and from different perspectives. According to the American Heritage Dictionary (https://ahdictionary.com), it can refer to a relationship in which there is correspondence or similarity between entities or parts, or invariance under transformation. This concept was also elaborated upon by Philip W. Anderson, Nobel laureate in Physics, who said: "**By symmetry we mean the existence of different viewpoints from which the system appears the same**" and that "it is only slightly overstating the case to say that physics is the study of symmetry" [44, p. 394]. We adopt Anderson's definition in this paper.

4.4 Symmetry as a Universal Property

In this section, we highlight symmetries in various domains, including in the natural world, in mathematics, in the laws that govern the universe, and in social interactions, aesthetics, and preferences.

4.4.1 Symmetries in the Natural World

In the natural world, symmetries can be highly visual. Flowers, plants and sessile animals (such as sea anemones) often have radial or rotational symmetry. The body shapes of humans and the major group of animals such as insects, fish, birds, and mammals, have bilateral symmetry. The forms of the majority of heavenly bodies such as the Earth, the sun, and galaxies, are symmetric. Other examples of symmetric objects or patterns include, snowflakes, sand dunes, waves, animal gait, animal markings, stalactites and stalagmites in caves—indeed, symmetries appear everywhere, and seem to appeal strongly to our innate sense of pattern [45]–[47].

4.4.2 Symmetries in Mathematics

In geometry, an object is symmetric if there is a transformation that moves individual parts of the object without changing the overall shape. Various types of symmetries have been studied in geometry, including: reflectional (mirror) symmetry (examples of which are the bilateral symmetry of human bodies and faces); rotational symmetry (examples of which include the fivefold symmetry of a starfish, or the sixfold symmetry of a snowflake); and translational symmetry (such as the traces of footprints left in the sand by a person walking on a beach).

In logic, a binary relation can be either symmetric or asymmetric. An example of a symmetric relation is: If A is a sibling of B then B is also a sibling of A. Symmetric logical operators include AND, OR, XOR, and $\leftrightarrow$ (if and only if).

Today's *formal* (mathematical) concept of symmetry came from algebra when the notion of a *symmetric group* [5] emerged, which led to the development of diverse areas of mathematics. The discovery of the connections between geometry and group theory (in particular, the group of transformations) allowed for theorems to be transferred between different areas of geometry [45] [49].

In mathematics, the concept of *isomorphism* [50, p. 3] expresses the idea that two mathematical objects have the same structure with respect to the properties under consideration. An isomorphism is therefore an instance of symmetry, but symmetry is not limited to isomorphisms or mathematical structures. The examples of Sections 3.1 and 3.2 are better understood as instances of symmetry than isomorphism.

5. Note the difference between a *symmetric group* and a *symmetry group*: The former is the group of permutations of n distinct objects—of order $n!$, whereas the latter is a group of symmetry-preserving operations, including rotations, reflections, and inversions [48].

4.4.3 Symmetries in Nature's Laws

At a fundamental level are the laws of nature, and one of the most intriguing features of these laws is that they are symmetric [45], [51], [52]. It should be noted that symmetry of the laws does not necessarily imply symmetry of the behavior: There is a difference between the symmetries of the "state" of a system, and the symmetries of the rules defining it [52]. It has been observed that the laws of nature are more symmetric than nature itself [45].

For example, Noether discovered the crucial relationship between symmetry and conservation [53]. Einstein wrote about Noether as "One seeks the most general ideas of operation which will bring together in simple, logical and unified form the largest possible circle of formal relationships." [46, p. 114]. Today, symmetry plays a major role in the search to unify relativity and quantum theory [45].

4.4.4 Other Symmetries

Symmetry does not need to be visual. Consider two children playing rock–paper–scissors [45]. It is symmetry that makes the game fair and attractive: Both players and all three strategies are on equal footing. Similarly, in social disputes, both sides should be treated in the same way [45]. Symmetrical social interactions, which may include asymmetrical balance, can be identified in a variety of contexts, including reciprocity and sympathy.

Symmetry is associated with our innate sense of beauty and harmony, and thus with aesthetics and preference. For example, it has been reported that symmetry in the human face and body correlates with attractiveness [54] and that movie stars often have unusually symmetrical faces [45]. Symmetry can be found in many artistic areas, such as in architecture, music, literature, and in the design of crafts and objects.

4.5 Symmetry as a Metamorphic Relation Pattern

We have seen that symmetry is an intrinsic, pervasive, and profound property of the real world. Computer systems are built to solve problems or meet requirements from the real world, often by simulating, modeling, or learning from it (including nature and society) or its processes. Computer software is designed and used by humans, whose preference and innate sense of harmony is often associated with symmetry [45], [47]. Therefore, we hypothesize that *symmetry is also a pervasively desirable property for many computer systems*.

Furthermore, there is a growing awareness that the concept of symmetry can serve as a heuristic to help scientists in their exploration of the unknown. For example, guided by Noether's theorem, physicists can now guess what the action might be when exploring the nuclear and subnuclear world [46]. Similarly, later in this paper, we show how the concept of symmetry (the *symmetry metamorphic relation pattern*, to use our terminology) can help a user to understand the behavior

of a computer system in the absence of a detailed system specification. We first give the following definition:

Definition 1: A *metamorphic relation pattern* (MRP) is an abstraction that characterizes a set of (possibly infinitely many) metamorphic relations.

To apply *abstraction* [55, p. 49], we need to identify the most important aspects of the MRs under consideration, ignoring details. One approach to the identification and grouping of MRPs (and MRs) is through *levels of abstraction*. In other words, it is possible for many MRPs to form a *hierarchy* [55, pp. 79-81], with MRPs at higher levels being more abstract, and those at lower levels being more concrete. Consider mathematical programs, for example. The property *monotone* can be considered an MRP. Compared with monotone, "strictly increasing" and "strictly decreasing" can be considered MRPs at a lower level of abstraction, and could have other MRPs further below them (for example, by considering linearity, or the shape of curvature). As another example, *symmetry* is an MRP at a high level of abstraction, and "equivalence under geometric transformation" would be an MRP under symmetry, at a lower level. We define the symmetry MRP as follows:

Definition 2: The *symmetry MRP* refers to the existence of different viewpoints from which the system appears the same.

Definition 2 borrows from Anderson's notion of "symmetry" (see Section 4.3), but here the word "system" can refer to not only a physical system, but also to a computer system. Using this definition, it is clear that the examples of Sections 3.1, 3.2, and 4.2 are all instances of the symmetry MRP. For example, regardless of what query language is being used, the meaning of the query ("American inventors" or "I love to eat apples") should be preserved, and from this perspective, the system should "appear the same." However, the users found that this was not the case; therefore, they can decide whether to accept the outputs, or to use alternative ways to get more suitable results.

In Definition 2, "the system appears the same" does not mean that the software system's (source and follow-up) outputs must have an equality or equivalence relation. This point is further elaborated in Section 7.

4.6 Metamorphic Relation Input Pattern (MRIP)

When defining the symmetry MRP, we did not specify the types of transformations that could be performed on the source input, or the types of invariance that should be observed when checking the output. An attempt to do so would weaken the generality of the MRP. However, to help practitioners to more easily identify concrete MRs, in this paper we propose a metamorphic relation input pattern (MRIP), *change direction*, as an approach to transforming the source input into the follow-up input.

Definition 3: A *metamorphic relation input pattern* (MRIP) is an abstraction that characterizes the relations among the source and follow-up inputs of a set of (possibly infinitely many) metamorphic relations.

As with MRPs, MRIPs are abstractions of relations. Therefore, the discussion about *levels of abstraction* and *hierarchy* under Definition 1 is equally applicable to MRIPs. For example, "change sequence," which changes the sequence of some elements in the source input, can be considered an MRIP at a high level of abstraction. Under this MRIP, there could be two more MRIPs at a lower level: "change sequence in time" and "change sequence in space," where the former could refer to the change of time sequence of some input events (such as in the context of testing an interactive system), and the latter could refer to the change of order of some data items (such as permuting an input array in the context of testing a sorting program). Typical operations such as "change value," "insertion," and "deletion" could all be used to create MRIPs at different abstraction levels. We can now define the "change direction" MRIP at a high level of abstraction as follows:

Definition 4: The *change direction* MRIP refers to the existence of a *direction* element in the source input, either physical or logical, explicit or implicit, which can be changed to construct the follow-up input.

The "change direction" MRIP can help the users and testers to identify "different viewpoints" of a system. The multiplicity of the association between "MRIP" and "MRP" is $n : n$, that is, an MRIP may be associated with multiple MRPs, and an MRP may also have multiple MRIPs. This means that the "change direction" MRIP may also be associated with an MRP other than symmetry, and vice versa. The example in Section 3.2 belongs to both the symmetry MRP and the "change direction" MRIP because there is a "direction" element in the source input ("Chinese to English") that was changed to "English to Chinese" in the follow-up input. In contrast, although the example in Section 3.1 belongs to the symmetry MRP, it does not belong to the "change direction" MRIP because there is no "direction" element in the MR. The MR "swapping the origin and destination nodes should not affect the length of the shortest path" (Section 2) is also an instance of both the symmetry MRP and the "change direction" MRIP.

In the rest of this paper, the "change direction" MRIP, in combination with the symmetry MRP, is applied to a variety of application domains. In Section 5, we test 65 top commercial websites by looking at their most commonly used features, *search* and *sort*. The "direction" element in the user input is the sort criterion (sort from high to low, or from low to high), and the symmetry is that sorting the results in ascending and descending orders should return the same set of results in reverse order. In Section 6, we test the Google Maps navigation service, where the "direction" element is the direction from the origin to

the destination, and the symmetry is that swapping the origin and destination should return a route that has a similar cost (in terms of time or distance, provided that no one-way restriction is involved). In Section 7, we test the Google Maps location-based search, where the "direction" element is the direction from one location to another location, and one of the symmetries is that if an entity A can see another entity B, then, everything else being equal, B can also see A. In Section 8, we test the face recognition functions of MATLAB, OpenCV, and Facebook, where the "direction" element is the direction of the x-axis of 2D images, and the symmetry is that changing the direction of the x-axis of an image from rightward to leftward (which results in a mirror image) should not affect the face recognition outcome because human faces usually have approximate bilateral symmetry [45], [47]. In Section 9, we test the Google video analysis service Cloud Video Intelligence, where the "direction" element is whether a video is played forwards or backwards, and the symmetry is in terms of time: The same static objects in the video should be identified regardless of whether the video is played forwards or backwards. We show that none of the systems under test satisfied these symmetry properties, thus revealing various issues—some of these issues are software faults; others reveal differences between the users' expectations and the actual behavior of the software. The detection of these issues emphasizes the need for developing user-side countermeasures.

5 CASE STUDY OF 65 TOP COMMERCIAL WEBSITES

In this section, we report on applying the symmetry MRP and "change direction" MRIP to commercial websites.

Unlike previous research on MT, where multiple MRs were identified together with a large number of test cases to test the SUTs [28], in this case study we tested the SUTs using only *one* MR and only *one* MT test case (a source test case and a follow-up test case) to identify software issues and corresponding user countermeasures. We considered the most commonly used features of online stores and advertising websites: *search* and *sort*. In particular, because research has shown that one of the most frequently used features by end users of commercial websites is *price sorting*, our study focused on this feature. Collins et al. [56, p. 68], for example, when studying the sorting strategies of end users, reported that "price is clearly the dominant sort attribute." Degeratu et al. [57, p. 75] studied how search attributes affected online choice behavior, and noted that "many executives are very concerned that online consumers will focus on price." In general, customers shopping online may face an overwhelming amount of information. An online search, in combination with sorting, enables them to "find the products and prices that best meet their needs," and to "both expand and narrow the consideration sets" [57, p. 77]. Suk et al. [58]

reported that the order in which options are presented could significantly influence consumer responses. They found, for example, that "when differing brand options are presented in descending price order, consumers tend to choose higher-price options; when they are presented in ascending price order, consumers tend to choose lower-priced options (the price order effect)" [58, p. 708]. It is therefore possible for companies to implement a pricing strategy to maximize benefits: a default list presenting items in descending order of price, for example.

In short, price sorting is a simple, but critical, function for both customers and owners of online stores and advertising websites.

5.1 The Metamorphic Relation

We define the following MR from the users' perspective, as a concrete instance of the "change direction" MRIP and the symmetry MRP, where the "direction" element is whether the sorting is from low to high, or from high to low:

$MR_{PriceSort}$: Let $s(q, c)$ denote a *perfect* search function, where q is a search criterion and c is a sorting criterion—the results are sorted according to c. If c_1 (c_2) denotes the criterion that sorts the results by price in ascending (descending) order, then $s(q, c_1)$ and $s(q, c_2)$ should return exactly the same set of results, but in reverse order.

It should be noted that $MR_{PriceSort}$ does not belong to the "equivalence" MROP proposed by Segura et al. [39], which does not consider the ordering of the search results.

5.2 Websites Under Test

We selected a total of 65 top commercial websites from several major ranking lists based on annual revenue or popularity. Of these 65 sites, 35 were top retailing websites listed in Deloitte's annual Global Powers of Retailing report [59] (which identified the biggest e-retailers in terms of annual revenue in 2015). We also selected 11 top pharmacy websites, based on web traffic, from the Alexa rankings [60]. We further selected 11 top real estate websites identified by eBizMBA, an eBusiness guide [61] (whose rankings were based on "each website's Alexa Global Traffic Rank, and U.S. Traffic Rank from both Compete and Quantcast."). Finally, similar to the real estate websites selection, we selected eight top car sales websites from the eBizMBA rankings [62]. We only selected English language websites whose features included price sorting.

Each website was tested against $MR_{PriceSort}$ by running *one* source test case and *one* follow-up test case. For each website, the source and follow-up test cases used exactly the same query term together with a request to sort by price in ascending and descending orders, respectively. Although it would have been desirable for all 65 websites to be tested using the same query term, this, however, was not possible due to the different nature of the websites. For example, real estate websites required the user to

enter location information of the real estate properties but a body care retailer did not require such information. Our approach, therefore, was to group the 65 websites into nine categories, as shown in the first column of Table 1. Websites within the same category were tested using the same query term (listed in the second column of Table 1). For example, all body care retailer websites were tested using the query term "body," and all office supply websites were tested using the query term "pencil." These query terms were generated manually such that they were general enough to accommodate all websites within the same category, and return nonempty search results. During testing, some websites did not return a results page for a general query term. For example, the website www.sears.com showed only a promotion page when the query term was "women." For those websites that could not return a results page, we used a backup query term ("women dress" instead of "women"). Similarly, we also prepared a backup query term for the category of online real estate websites. These two backup query terms are shown in the third column of Table 1.

All 65 websites under test and their corresponding categories are listed in the first and second columns of Table 2.

TABLE 1: Query terms.

Category	Query	Backup query
body care retailer (1 website)	body	
car sales (8 websites)	BMW (zip:11223)	
consumer electronics retailer (2 websites)	mouse	
department retailer (24 websites)	women	women dress
groceries (3 websites)	sauce	
home improvement (3 websites)	knife	
office supply (2 websites)	pencil	
online pharmacy (11 websites)	oil	
online realestate (11 websites)	Brooklyn, NY	New York, NY

5.3 Analyses of Test Results

Only *five* of the 65 websites passed all our tests, as highlighted in rows #14, #31, #38, #45, and #52 of Table 2.

We categorized the software issues that caused the MR violations into five types, as listed in columns #3 to #7 of Table 2. In the following, we analyze these five types of issues, discuss their impact on the software quality characteristics of commercial websites (referring to the software quality model standard ISO/IEC 25010 [63]), and explain how the use of MRs can help users to better achieve their goals.

### 5.3.1 Type 1 Issue: Count Consistency (Functional Correctness)

A characteristic of $MR_{PriceSort}$ is that the number of results in the source and follow-up outputs should be equal, otherwise at least one of the two outputs must be incorrect. Count consistency, therefore, is related to the *functional correctness* of the SUT, which refers to the "degree to which a product or system provides the correct results with the needed degree of precision" in ISO/IEC 25010 [63].

5.3.1.1 **Observations**: Column #3 of Table 2 shows the count consistency result for each website under test. A tick (✓) indicates a pass (consistent); and a cross (×) indicates a failure (inconsistent). Table 2 shows that, of the 65 websites under test, five (7.69%) failed to produce consistent counts. For example, when searching for "women dress" in www.amazon.com, 160 results were returned when sorted by price (low to high) (see Fig. 3a), but 851,077 were returned when sorted by price (high to low) (see Fig. 3b). We note that, in the results page, there is a statement "Showing most relevant results. See all results for women dress." In this example, our focus was on the "most relevant results;" however, even when we clicked on the link "See all results for women dress," the two result counts were still different. Theoretically, the difference could be caused by web dynamics such as database updates. To exclude this possibility, we *repeated the source and follow-up tests ten times*, involving a total of 20 queries as follows: sort by price (low to high); sort by price (high to low); sort by price (low to high); . . .. In all of the (low to high) queries, "160 results" were returned, and in all of the (high to low) queries, "851,077 results" were returned. This means that the inconsistency was not caused by the dynamic nature of the website. Furthermore, we repeated these queries on different days, obtaining similar results. This means that the inconsistency was not caused by the use of temporary, cached data, or by any similar mechanism in which the server might simply return the same result for the same query without recalculation.

In addition to the total counts, we further found that the result counts for subgroups could also be inconsistent. For example, when searching for "women" in www.target.com, the counts of four size groups were inconsistent: the "woman" group (5,746 vs 5,737); the "juniors" group (1,977 vs 1,982); the "plus" group (990 vs 984); and the "maternity" group (509 vs 510), as shown in Fig. 3c. We repeated these tests multiple times, always obtaining the same result, which means that the MR violation was not caused by data updates in the website.

5.3.1.2 **Discussion**: We reiterate that, in this paper, *all reported inconsistencies were repeatable at the time of the experiment, thus excluding the possibility that the MR violation was caused by data updates on the server side.*

One may argue that some websites might only return approximate results and, therefore, imprecision should be allowed. In the case of the Amazon failure, it is also possible that different search or approximation algorithms were used for the sorting criteria "low to high" and "high to low." Third-party components running on different servers might have also been involved, thus causing the inconsistency of the search results shown in Figs. 3a and 3b. Without further knowledge of the system design, we are unable to confirm the cause of the failure, but developers who know more about the system design should be able to find the root cause, and rectify the problem. Readers who are interested in the validity of MR use for verification, validation, and quality assessment of
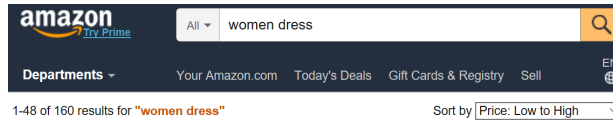
(a) A source query: search for "women dress" in www.amazon.com, sorted by price (low to high)—160 "most relevant" results.
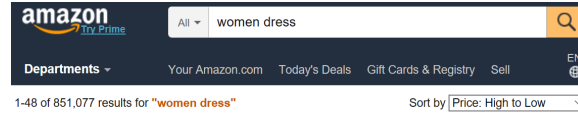
(b) A follow-up query: search for "women dress" in www.amazon.com, sorted by price (high to low)—$851,077$ "most relevant" results.
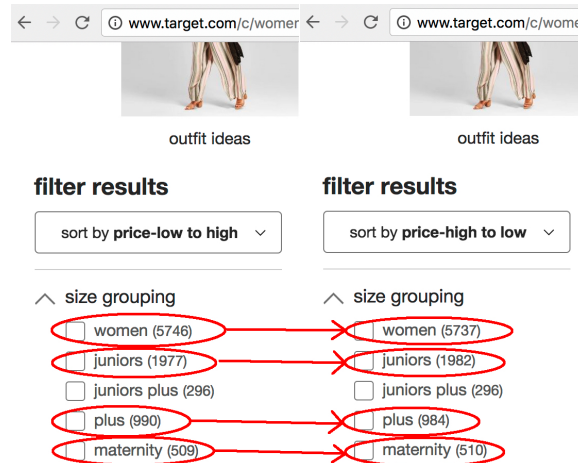
(c) Inconsistent results identified by comparing the source (left) and follow-up (right) outputs using the MR, when searching for "women" in www.target.com.

Fig. 3: Examples of Type 1 issue: failures detected in www.amazon.com and www.target.com using a metamorphic relation.
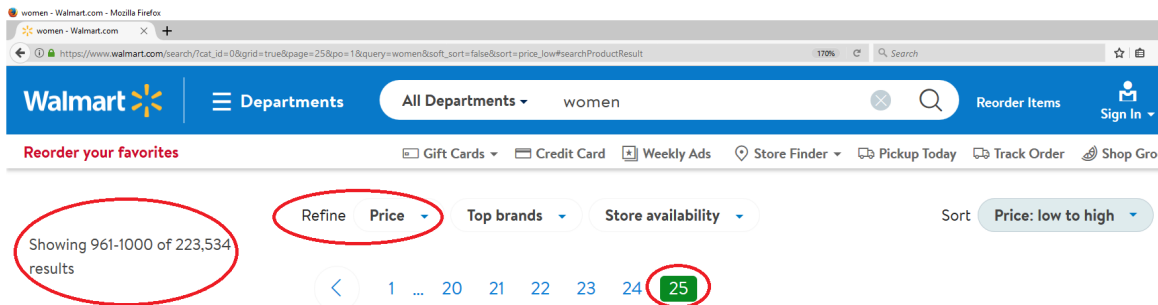
Fig. 4: Example of Type 2 issue: Walmart can only display up to 1000 records.

search services are referred to the in-depth discussions in our previous work [9], [38]. In this paper, however, our perspective is from that of the *users*. After all, the online systems are designed to satisfy the users' needs, and the users do not know the technical design details of the systems—they only want to know whether the system is suitable for their information needs. Consider the example shown in Figs. 3a and 3b. This type of inconsistency in counting the "most relevant" results could confuse the users, probably failing to meet their expectation and, hence, failing the validation.

Nevertheless, the users are now in a *better position* to decide what to do next. This is because the MR yielded useful information from comparing the follow-up output (Fig. 3b) with the source output (Fig. 3a), and hence the users can better understand how Amazon works—they may immediately realize that different sorting criteria have a direct impact on the search results. The users can

TABLE 2: Results of experiments (continued on next page).

	Website	Category	Type 1 count consistency	Type 2 completeness of results	Type 3 no separate sections	Type 4 same price reverse	Type 5 different prices reverse
1	www.bathandbodyworks.com	body care retailer	✓	✓	✓	×	✓
2	www.autotrader.com	car sales	✓	✓	×	×	✓
3	www.carfax.com	car sales	✓	×	×	×	✓
4	www.cargurus.com	car sales	✓	✓	×	×	✓
5	www.carmax.com	car sales	✓	✓	✓	×	✓
6	www.cars.com	car sales	✓	×	×	×	×
7	www.carsdirect.com	car sales	✓	×	×	×	✓
8	www.kbb.com	car sales	✓	×	×	×	✓
9	www.thecarconnection.com	car sales	✓	✓	×	×	✓
10	www.bestbuy.com	consumer electronics retailer	✓	✓	✓	×	✓
11	www.currys.co.uk	consumer electronics retailer	✓	✓	✓	×	✓
12	www.amazon.com	department retailer	×	×	×	×	×
13	www.costco.com	department retailer	✓	✓	✓	×	✓
14	**www.hsn.com**	**department retailer**	✓	✓	✓	✓	✓
15	www.jcpenney.com	department retailer	✓	✓	✓	×	✓
16	www.johnlewis.com	department retailer	✓	✓	✓	×	✓
17	www.joybuy.com	department retailer	✓	✓	✓	×	×
18	www.kohls.com	department retailer	✓	✓	✓	×	×
19	www.llbean.com	department retailer	✓	✓	✓	×	×
20	www.macys.com	department retailer	✓	✓	✓	×	×
21	www.overstock.com	department retailer	✓	×	✓	×	×
22	www.qvc.com	department retailer	✓	✓	✓	×	✓
23	www.sears.com	department retailer	✓	×	✓	×	✓
24	www.target.com	department retailer	×	×	✓	×	×
25	www.very.co.uk	department retailer	✓	✓	✓	×	✓
26	www.walmart.com	department retailer	×	×	✓	×	✓
27	www.wayfair.com	department retailer	✓	✓	✓	×	×
28	bedfordfair.blair.com	department retailer	✓	✓	✓	×	×
29	shop.nordstrom.com	department retailer	✓	✓	✓	×	×
30	store.nike.com	department retailer	✓	✓	✓	×	×
31	**www.asos.com**	**department retailer**	✓	✓	✓	✓	✓
32	www.hm.com	department retailer	✓	✓	✓	×	✓
33	www.neimanmarcus.com	department retailer	✓	✓	✓	×	✓
34	www.next.co.uk	department retailer	✓	✓	✓	×	✓
35	www.zalando.co.uk	department retailer	✓	✓	✓	×	✓

TABLE 2: Results of experiments (continued).

	1	2	3	4	5	6	7
	Website	Category	Type 1 count consistency	Type 2 completeness of results	Type 3 no separate sections	Type 4 same price reverse	Type 5 different prices reverse
36	www.ocado.com	groceries	✓	✓	×	×	✓
37	www.sainsburys.co.uk	groceries	✓	✓	✓	×	✓
38	**www.tesco.com**	**groceries**	✓	✓	✓	✓	✓
39	www.habitat.co.uk	home improvement	✓	✓	✓	×	✓
40	www.homedepot.com	home improvement	✓	×	✓	×	×
41	www.williams-sonoma.com	home improvement	✓	✓	×	×	✓
42	www.staples.com	office supply	✓	×	×	×	✓
43	www.viking-direct.co.uk	office supply	✓	✓	×	×	×
44	medichest.com	online pharmacy	✓	✓	✓	×	✓
45	**mexmeds4you.com**	**online pharmacy**	✓	✓	✓	✓	✓
46	well.ca	online pharmacy	✓	✓	×	×	✓
47	www.cincottachemist.com.au	online pharmacy	✓	✓	✓	×	×
48	www.cvs.com	online pharmacy	✓	✓	✓	×	✓
49	www.epharmacy.com.au	online pharmacy	✓	✓	✓	×	✓
50	www.medshopexpress.com	online pharmacy	✓	✓	✓	×	×
51	www.netpharmacy.co.nz	online pharmacy	✓	✓	✓	×	✓
52	**www.pharmacy2u.co.uk**	**online pharmacy**	✓	✓	✓	N/A	✓
53	www.pharmcom.com	online pharmacy	✓	✓	✓	×	✓
54	www.walgreens.com	online pharmacy	✓	✓	×	×	✓
55	hotpads.com	online realestate	✓	×	✓	×	×
56	www.apartmentguide.com	online realestate	✓	✓	✓	×	×
57	www.apartments.com	online realestate	×	×	×	×	✓
58	www.homes.com	online realestate	✓	✓	✓	×	✓
59	www.realtor.com	online realestate	✓	✓	✓	×	✓
60	www.redfin.com	online realestate	✓	×	✓	✓	✓
61	www.remax.com	online realestate	✓	✓	✓	×	✓
62	www.rent.com	online realestate	✓	✓	×	×	×
63	www.trulia.com	online realestate	✓	×	✓	×	✓
64	www.zillow.com	online realestate	×	×	✓	×	✓
65	www.ziprealty.com	online realestate	✓	✓	✓	×	✓
	Inconsistency rate (%)		7.69	26.15	24.62	90.77 (=59/65)	29.23

13

now decide to accept the 160 "most relevant" results in the source output, if they prefer fewer results; or the $851,077$ "most relevant" results in the follow-up output, if they prefer a larger dataset.

From the users' perspective, the use of the MR has helped to improve the system's *appropriateness recognizability* ("degree to which users can recognize whether a product or system is appropriate for their needs") and *learnability* ("degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use") [63]. Both *appropriateness recognizability* and *learnability* are software quality subcharacteristics under the software quality characteristic of *usability* in ISO/IEC 25010 [63]. For users who need accurate and stable results, such as for market research, the MR allows them to quickly recognize that Amazon might not meet their information needs, and hence they might decide to consult other websites.

Finally, the "degree of precision" is a critical notion in the definition of "functional correctness" [63]. The results presented in column #3 of Table 2 mean that the MR can help users to distinguish between high-precision and low-precision outputs.

5.3.2 Type 2 Issue: Completeness of Search Results (Functional Completeness, Capacity, Usability)

Another characteristic of $MR_{PriceSort}$ is that the source and follow-up outputs "should return exactly the same set of results." Some websites failed to do this and, therefore, violated the MR. These websites only returned partial results (rather than a complete set of results) when the number of results was large, and different partial results were returned when sorted in ascending and descending orders.

Incompleteness of the search results may not necessarily be related to the functional correctness, but is related to the *functional completeness*, *capacity*, and *usability* of the websites under test, where functional completeness refers to the "degree to which the set of functions covers all the specified tasks and user objectives"; capacity refers to the "degree to which the maximum limits of a product or system parameter meet requirements"; and usability refers to the "degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [63].

5.3.2.1 **Observations**: Column #4 of Table 2 shows that, of the 65 websites under test, 17 (26.15%) failed to provide complete results, which caused violations of $MR_{PriceSort}$. For example, we visited the Walmart website (www.walmart.com) and entered the query term women to search. Under "Sort," we selected "Price: low to high." In the search results page, the system stated:

> Showing results sorted by both Relevance and Price: low to high.
> Sort results by Price: low to high only.

We clicked on the link "Sort results by Price: low to high only" to sort by price only, and the system reported that "$223,534$ results" were found. However, when we jumped to the last search results page (which was the 25th page, where each page contained 40 results), we could only find a total of 1000 results, as shown in Fig. 4. In other words, the results list was truncated to show only the top 1000 items. The ascending and descending lists thus truncated, of course, contained different items, hence violating $MR_{PriceSort}$. The 17 failed websites imposed different threshold values (the maximum sizes of the results lists) when applying truncation, ranging from a few hundred items to tens of thousands of items.

5.3.2.2 **Discussion**: Truncating the results list could be a valid design decision based on practical considerations. It is a strategy adopted by many web search engines, including Google [64], and this practice could be justified by the observation that search engine users are normally only interested in the top few results [9]. Nevertheless, there are still criticisms of this practice because, according to user behavior studies at Blekko [64], "there are people who look deep into the results."

It should be noted that the 65 websites under study were specific commercial sites rather than generic web search engines, and that in our experiments the results were ranked **by price rather than by relevance**—users might not be interested in web pages of lower relevance, but they may be extremely interested in products of lower or higher prices. The argument that "users are normally only interested in the top few results" would only be valid when sorting by relevance: Here the argument is invalid, as we were sorting by price. The 17 websites' practice of excluding higher (or lower) priced items from their search results is therefore misleading.

The problem of these 17 websites can be demonstrated using the example in Fig. 4: When sorting by price, Walmart only shows the top 1000 of the $223,534$ products. This means that the users can only see the 1000 least expensive products (when sorted in ascending order) or the 1000 most expensive products (when in descending order). The vast majority of the products ($223,534 - 1000 - 1000 = 221,534$) in the median price range are hidden. A crucial deficiency in the 17 websites' *functional completeness* and *capacity* has therefore been revealed.

A further *usability* problem is that the users can be easily misled to consider the last displayed item as the most (or least) expensive product, unaware of the fact that there are actually far more available products not being displayed.

Using this MR, users will find that the top results in an ascending list are completely different from the bottom results in a descending list, which may prompt them to further investigate the results pages, where they would find that both lists are actually incomplete. The users could then decide whether they should change their search or browsing strategy. They may, for example, perform an additional operation, such as "Refine Price" (as shown in Fig. 4), which allows users to find products

TABLE 3: Additional backup query terms for studies conducted in Sections 5.3.3, 5.3.4, and 5.3.5.

Category	1st backup query	2nd backup query	3rd backup query
body care retailer	soap		
car sales	BMW 328i (zip:11223)	BMW 3 series Gran Turismo (zip:11223)	BMW 3-series (zip:11223)
consumer electronics retailer	laptop		
department retailer	women dress	percolator	
groceries	oil		
home improvement	cement tile		
office supply	Dell all in one printer		
online pharmacy	zinc		
online realestate	zip:11223		

```
Begin
Issue a source query using Table 1.
If the results list has been truncated then
  Issue a source query using the "1st backup query" in Table 3. For
    websites where the "1st backup query" is invalid, use the "2nd
    backup query." For websites where both the "1st backup query" and
    the "2nd backup query" are invalid, use the "3rd backup query."
  If the results list has been truncated then
    Issue the source query again together with a filter, such as choosing a
      color, a size group, a department, or a category (different websites
      offer different filters).
  EndIf
EndIf
End
```

Fig. 5: Procedure for generating a source query and avoiding result truncations. This procedure was followed in the studies reported on in Sections 5.3.3, 5.3.4, and 5.3.5.

within a specific price range. It should be noted, however, that "Refine Price" could alleviate, but not completely solve the problem because, for example, even the minimal price range in the Walmart website could easily contain more than 2000 results, and yet the system could not show more than 1000. To further alleviate this problem, the users could apply more filters, such as restricting the product types or departments.

5.3.3 Type 3 Issue: Separate Sections (Functional Completeness, Functional Appropriateness)

A further characteristic of $MR_{PriceSort}$ is that the source and follow-up outputs must be "in reverse order." Some websites failed to meet this requirement because they inserted separate sections into the results lists that violated the ascending and descending orders. More specifically, we found that some items were always at the beginning or the end of the results lists, regardless of their prices. Therefore, we say that these items belong to *separate sections*, independent of price sorting. Consequently, users could not directly find the lowest or highest price by just looking at the top or the bottom of the "sorted" list—additional effort would be needed for the users to identify which part of the results list was sorted and which part was not.

The insertion of the "separate sections" is normally a design decision and, therefore, may not necessarily imply a bug. However, this does adversely affect the *functional completeness* and *functional appropriateness* of the websites, where the concept of functional completeness was introduced in Section 5.3.2, and functional appropriateness refers to the "degree to which the functions facilitate the accomplishment of specified tasks and objectives"—an example is that "a user is only presented with the necessary steps to complete a task, excluding any unnecessary steps" [63].

5.3.3.1 **Additional experimental settings**: In Sections 5.3.3, 5.3.4, and 5.3.5, we further examine the actual contents of the results lists. To be fair to all of the websites under study, we wanted to avoid the "truncation" effect discussed in Section 5.3.2. Therefore, we manually created an additional set of backup query terms, as shown in Table 3. If the original query terms from Table 1 resulted in truncation of the results list, a backup query term from Table 3 was used instead, with further constraints also possibly added. The detailed algorithm, which effectively eliminated the truncation effect, is shown in Fig. 5.

5.3.3.2 **Observations**: Column #5 of Table 2 is titled "no separate sections." A cross ($\times$) indicates the occurrence of separate section(s) that caused an $MR_{PriceSort}$ violation. Of the 65 websites under test, 16 (24.62%) yielded separate sections that violated $MR_{PriceSort}$.

5.3.3.3 **Discussion**: Some websites tended to put featured or promotional products at the beginning of the results list and/or leave low priority items (such as those out of stock) to the end of the list, regardless of the sorting criteria. Although this strategy could provide benefits to some vendors or customers, it should be noted, however, that this practice resulted in a functional deficiency: The system could not generate a truly sorted list of results according to user criteria.

From the perspective of user experience, having separate sections can be both good and bad: It is good, for example, for users who are interested in promotions, or not interested in unavailable products; it is bad, however, for users who want to obtain a list of items sorted by price, regardless of the promotional or availability status of the items (for example, researchers doing market studies, or consumers making general price comparisons).

From the users' perspective, by issuing a follow-up query to sort the results the other way round, they should be able to more easily identify the separate sections—because these sections appear in the same locations regardless of the search criteria.

To improve user experience and software quality in terms of functional completeness and functional appropriateness, the website developers may consider providing an additional option for users. Such an option should allow the generation of a truly sorted list according to user criteria without separate sections.

5.3.4 Type 4 Issue: Same Price (Functional Completeness, Usability)

In Sections 5.3.4 and 5.3.5, we further examine the actual items contained in each search results list. A "search results list" contains multiple "search results pages." Some websites can return a very large number of search results, to be displayed in a large number of

search results pages. To improve testing efficiency, we decided to only check the first and last three search results pages when comparing the actual contents of the source and follow-up outputs. We used the websites' default page lengths (such as 60 results per page—different websites differed). Although this meant a weakened testing criterion, nevertheless, where there is no ambiguity, we still use $MR_{PriceSort}$ to denote the MR.

In $MR_{PriceSort}$, a plausible requirement for a *perfect* search function is that the order of the items having the same price in the source output should be reversed in the follow-up output. The majority of the websites under test failed to meet this requirement as they listed items of the same price in the same order in both the source and follow-up outputs—this was probably a design decision and, therefore, should not necessarily be considered a fault. However, this practice adversely affects the *functional completeness* and the *usability* of the websites, as will be explained shortly.

5.3.4.1 **Observations**: Column #6 of Table 2 is titled "same price reverse," which means that the order of the items with the same price in the source output should be reversed in the follow-up output. A tick (✓) indicates that the website satisfied this requirement, and a cross (×) indicates that it did not satisfy the requirement and, hence, violated $MR_{PriceSort}$. Of the 65 websites under test, 59 (90.77%) failed to satisfy this requirement. For the remaining six websites, one is labeled "N/A" (because its search results did not involve items of the same price), and *five* satisfied the MR by faithfully reversing the order of same-price items.

An example of a Type 4 issue that violated $MR_{PriceSort}$ is shown in Fig. 6: When searching for "knife" in http://www.williams-sonoma.com, and sorting the results by price (low to high), the system listed the "KAI for Williams Sonoma Paring Knife" ($7.95) before the "KAI for Williams Sonoma Citrus Knife" ($7.95) (Fig. 6a). The same order was followed when the results were sorted by price (high to low) (Fig. 6b). This was probably a result of the tie-break mechanism of the system and, hence, should not necessarily be considered a fault. However, there are problems with this, explained as follows: In a large commercial website, there can be thousands of products with the same price. For example, the Walmart website could not finish displaying all of its $19.99 products within its limit of 1000 records, for the query term "women." In general, the higher an item is ranked in the search results list, and the more frequently it appears, the more visitors it will receive [65]. That is why *search engine optimization* (SEO) is considered to be one of the most critical Internet marketing strategies, and vendors are extremely concerned about how their products rank in a search results list [66]. In our example, however, the tie-break mechanism (the reason why the "KAI for Williams Sonoma Paring Knife" always ranked higher than the "KAI for Williams Sonoma Citrus Knife") remains unknown to the
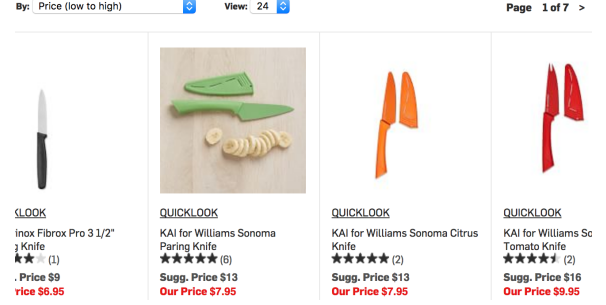
(a) Search for "knife" in www.williams-sonoma.com, sorted by price (low to high).
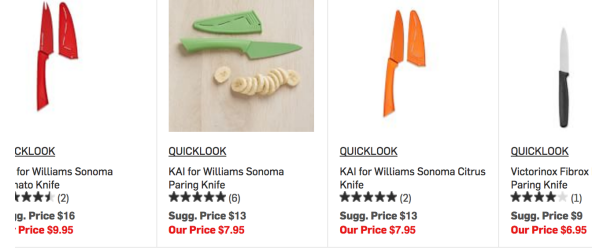

(b) Search for "knife" in www.williams-sonoma.com, sorted by price (high to low).

Fig. 6: An example of Type 4 issue that violated $MR_{PriceSort}$: The two $7.95 items were listed in the same order when sorted by price (low to high) and (high to low).

users (including both the consumers who buy knives and the vendors who sell them). This adversely affects the *appropriateness recognizability* and *learnability* of the system.

Another example of a Type 4 issue is given in Fig. 7, which shows the Walmart website displaying $10 products in the same order in both the ascending and the descending lists.

5.3.4.2 **Discussion**: Let $A1$, $B2$, $C3$, $D4$, $E4$, $F4$, $G5$, $H6$, and $I7$ denote products whose prices are $1, $2, $3, $4, $4, $4, $5, $6, and $7, respectively. Suppose the SUT has a Type 4 issue and returns lists $List_A$ and $List_D$ when sorting these items by price in ascending and descending orders, respectively: $List_A = (A1, B2, C3, D4, E4, F4, G5, H6, I7)$, $List_D = (I7, H6, G5, D4, E4, F4, C3, B2, A1)$. When examining each individual list separately, there is no problem, but if they are checked against the MR, then the following problems can be identified. First, the listing favors $D4$, because $D4$ always appears first among all the $4 products, regardless of how the products are sorted. In contrast, $F4$ is treated unfairly because it always ranks lower than $D4$ and $E4$. This is likely to increase the visibility of $D4$ (that is, give a higher exposure rate for customers to see $D4$) and decrease the visibility of $F4$. Visibility is a critical concern for online businesses. Worse than this, however, as explained in Section 5.3.2, the system may truncate results lists, which means that $F4$ has a higher chance than $D4$ of being
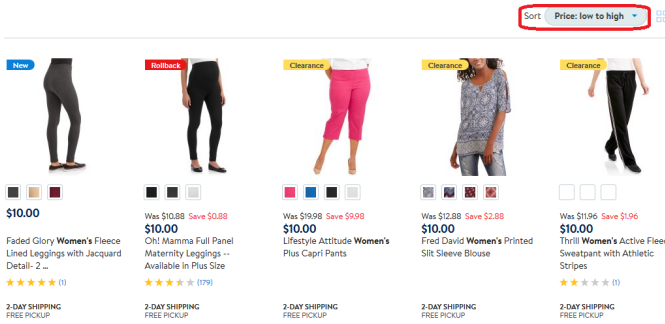
(a) Part of a search results page for $10 "women" products, sorted by price (low to high).
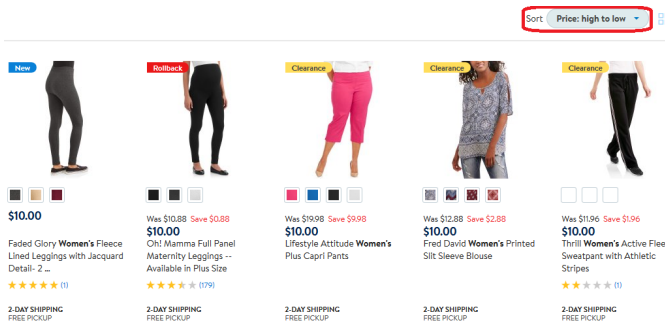
(b) Part of a search results page for $10 "women" products, sorted by price (high to low).

Fig. 7: An example of Type 4 issue at www.walmart.com: The five $10 products in both ascending and descending lists appeared in the same order.

truncated and never seen by customers, in both the ascending and descending order lists. Furthermore, the website's rule for sorting same-price items is hidden from users (both consumers and vendors), with no way for the users to set their preferred sorting rule—this is a functional deficiency.

The Type 4 issue may also be misleading when users are performing certain tasks. For example, consider the following scenario: A user requests the system to sort the results in ascending order. She then browses the results sequentially, seeing $A1$, $B2$, $C3$, $D4$. She stops at $D4$, and requests the system to sort the results again, but in descending order. She then browses the results in the new list, also sequentially, and sees $I7$, $H6$, $G5$, $D4$. When she reaches $D4$, she realizes that she has already visited this item when the list was in ascending order, and therefore concludes that she has already seen all of the results. She thus discards the list. Here, the user has been misled: She has never actually seen $E4$ or $F4$. This scenario shows that the Type 4 issue may lead to user errors, which is a major usability deficiency.

In general, the symmetry principle in social interactions introduced in Section 4.4.4 requires that different vendors should be treated fairly. The Type 4 issue violates this principle as some products are disadvantaged in terms of visibility. Our MR simply and explicitly asks the

users to compare the source and follow-up outputs, and hence allows the users to discover this situation. Mistakes like the one described in the preceding paragraph, can therefore more easily be avoided. Using the MR, vendors can also see that their products are advantaged or disadvantaged in terms of ranking and visibility.

5.3.5 Type 5 Issue: Different Prices (Functional Correctness, Usability)

While the Type 4 issue involved *same-price* items, the Type 5 issue refers to the $MR_{PriceSort}$ violation involving inconsistent sorting of *different-price* items. To study this type of issue, we ignored the "separate sections" discussed in Section 5.3.3 when testing the websites against $MR_{PriceSort}$.

It can be time consuming for users to identify incorrectly sorted items in a long results list; however, by generating both the source and the follow-up outputs, we found that out-of-order items could often be quickly detected when comparing only the top and bottom few results of the two lists. Two such examples are discussed in the following.

Fig. 8 shows excerpts of a screenshot from www.macys.com. The price sorting was erroneous, which was detected when we found that the top result (the AUD 9.43 product highlighted in the figure)[6] of the ascending list did not appear at the bottom of the descending list. This finding suggests that if users of this website want to browse all products in a particular price range, then they had better not use price sorting, because it could be wrong: To achieve their goal, users could use price filters instead of price sorting—here we are using another MR under the symmetry pattern because the activity of searching with a specified price range (price filter) and the activity of browsing a segment of a list sorted by price are theoretically equivalent.

An MR may also reveal the internal working mechanisms of the website under test. For example, Fig. 9 shows a Type 5 MR violation in www.amazon.com, detected by comparing the top result (the $7.99 product in Fig. 9a) of the ascending list with the bottom result (the $11.99 product in Fig. 9b) of the descending list. The MR was violated as the original $7.99 product in the ascending list was displayed as a $19.98 product in the descending list (Fig. 9c) and, hence, appeared in the middle of the list. A further investigation revealed that the item shown in Figs. 9a and 9c actually represented a *group* of products with different sizes for which the price ranged from $7.99 to $19.98—this was indicated by the phrase "See Size Options" (shown in the screenshots). In the source query, the user requested a sort by price (low to high), and the system used the lower bound ($7.99) to represent the entire group; in the follow-up query, the user requested sorting by price (high to low), and the system used the upper bound ($19.98) to represent the entire group.

---

6. The prices were shown in AUD as we visited this website from Australia.
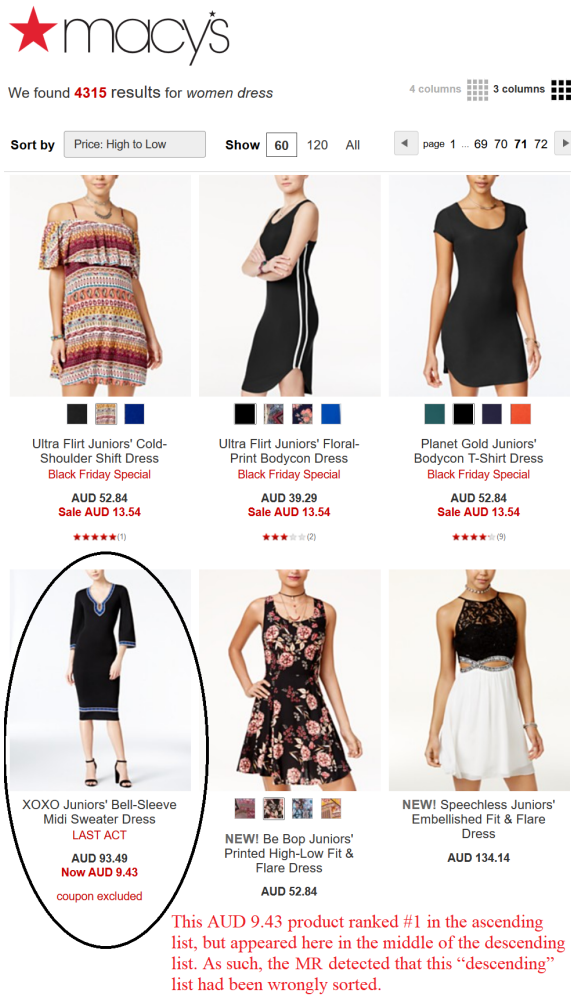
Fig. 8: Part of a results list wrongly sorted by price (high to low) at www.macys.com.
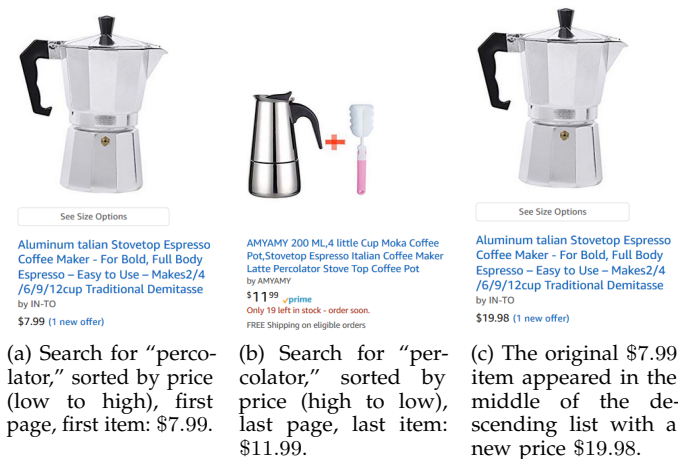

(a) Search for "percolator," sorted by price (low to high), first page, first item: $7.99.

(b) Search for "percolator," sorted by price (high to low), last page, last item: $11.99.

(c) The original $7.99 item appeared in the middle of the descending list with a new price $19.98.

Fig. 9: The MR revealed how products were priced in www.amazon.com.

This example shows that the MR allowed the user to understand how the Amazon system worked, and hence could adjust his/her search strategy accordingly.

This example also revealed a usability problem of the website, as most users might not realize that the displayed price was only an upper or lower bound of a product group—in this situation, developers should consider displaying a price range rather than a single price.

The test results are summarized in column #7 of Table 2: Type 5 MR violations have been revealed in a total of 19 (29.23%) websites.

5.4 Validity and User Experience

The connection between our approach and usability/user experience has been further validated through the following university laboratory exercise: A university postgraduate course in software engineering had a total of 36 students. These students were all asked to visit www.carsdirect.com and find both (i) the most expensive and (ii) the least expensive *used cars* satisfying the following criteria: Make: BMW; Model: 3-Series; Zipcode: 11223; and Distance (from zipcode area): 50 miles. The task was performed in a computer laboratory, with no time limitation set.

To our great surprise, all 36 students gave wrong answers. The majority (32 students[7]) reported $2495 to be the minimum price, because they had sorted the cars by price, from low to high, and then picked the first-ranked result (Fig. 10a). Similarly, these 32 students reported $38,995 to be the highest price, using the high to low listing (Fig. 10c). These answers were wrong because the lowest price was actually $1995 (Fig. 10b), and the highest price was actually $51,175 (Fig. 10d).

After this task, we taught the students the following MR concepts: The first MR was $MR_{PriceSort}$. The second MR also belongs to the symmetry MRP, and can be stated as follows: Suppose that we search a system for the most (or least) expensive product. Suppose the system returns product p with a price c. We can then issue a new query using the same query term together with a constraint $price > c$ (or $price < c$). A correct system should return an empty set.

Using the first MR, the students were able to improve their search results by looking at both the top and the bottom of both the ascending and the descending results lists to identify the lowest/highest prices. Using the second MR, the students were able to improve their results by recursively applying the price filter. For example, if the system returned $38,995 ($2495) as the most (least) expensive car then the students were able to issue a follow-up query by setting the price filter to be greater than $38,995 (or less than $2495). This process was repeated until no more results could be returned. Using either MR, the students were able to improve their search results.

The 100% error rate of the 36 computer science students finding the target cars, and the subsequent 100% improvement using MRs, demonstrate the validity
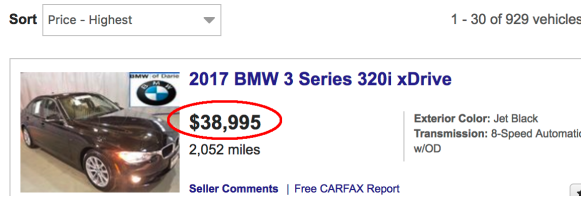
---

7. The remaining four students reported other results that were also incorrect.
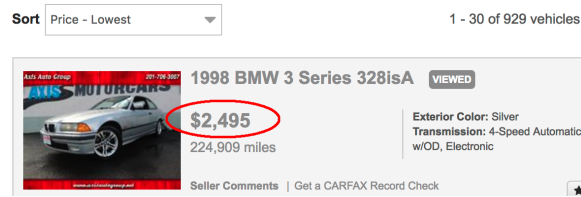
(a) Sorted by price (low to high), first page, first item—taken by most students as the cheapest car in the results list.

(b) This was the actual car with the lowest price in the results list.

(c) Sorted by price (high to low), first page, first item—taken by most students as the most expensive car in the results list.

(d) This was the actual car with the highest price in the results list.

Fig. 10: Students failed to find the target cars in www.carsdirect.com.

and efficiency of our approach for user experience improvement.

5.5 Summary

In contrast to the existing software testing literature, where large numbers of test cases are used by developers or testers to detect software issues, in this section we have shown how users can detect software issues using just one MR and two test cases. The MR studied is an instance of the combination of the symmetry MRP and the "change direction" MRIP. Our approach has revealed five types of issues in major websites. These issues include both software faults and other problems related to the functional correctness, functional completeness, functional appropriateness, and usability of the websites. The 65 websites under study are representative of large online businesses, and are listed in major lists of top sites according to popularity and revenue rankings.

Most importantly, based on the concept of the symmetry MRP, we have presented countermeasures that users can use to better achieve their objectives when dealing with these commercial websites whose internal working

mechanisms are hidden. The validity of our method has also been confirmed through an exercise involving end users.

6 CASE STUDY OF GOOGLE MAPS NAVIGATION

In this section, we report on a case study of applying the symmetry MRP and the "change direction" MRIP to a very different type of system: navigation software.

Navigation software attempts to generate an optimal route from an origin to a destination, conforming to the constraints given by the user. The route includes directions for the user to reach the destination. Not only is navigation software one of the most popular Internet applications, but it is actually the number one smart phone application, installed on more than 50%, globally [34]. Navigation systems are mission critical because errors in navigation may result in accidents, especially when such systems are used to guide self-driving cars, delivery robots, or autonomous drones.

Although navigation systems are both popular and critical, they are also very difficult to test because of the oracle problem. It is also challenging for a user to judge whether or not a route generated by the system is the best solution. In this section, we report on using the Google Maps web application at maps.google.com to show how our method can help users and testers. Google Maps is by far the most popular mapping service [34].

6.1 The Metamorphic Relation

In the context of navigation software, a "direction" element can be naturally identified as the direction from the origin to the destination. Therefore, the "change direction" MRIP can be applied in combination with the symmetry MRP to define the following concrete MR: *Swapping the origin and destination should return a route with a similar cost (provided that no one-way restrictions are involved in the routes)*. In this study, the *cost* of a route is in terms of distance or time (*either* similar distances *or* similar times will satisfy the MR). The "time" is the theoretical travel time without considering real-time data such as traffic jams. Financial costs are also not considered. It should be noted that this MR cannot be interpreted in any absolute sense, because special road conditions may exist. This MR is similar to the one for the shortest path problem discussed in Section 2.

6.2 Results

To more effectively avoid one-way restrictions in the routes, we selected walking, instead of driving, navigation to test in our experiments. This is because, according to our observations, there are far fewer one-way roads for pedestrians than for cars. All test results were manually validated. Readers who are interested in the testing of driving navigation without one-way restrictions are referred to our previous work [34], where four MRs

were identified to test driving navigation, three of which belong to the symmetry MRP.

In the present research, our experiments were conducted as follows: We selected London and Hong Kong as the subject cities. For both of these cities, we generated 1000 pairs of valid addresses. For each pair of addresses (A, B), we ran Google Maps to generate walking navigation from A to B and then from B to A. The distances and time durations of the returned routes were then compared. In all experiments, we used the default settings in Google Maps. As mentioned previously, to improve accuracy and make the experiments repeatable, no real-time traffic data were used. To avoid personalized results, we did not log into any online accounts, including those for Google.

For London, 8% and 42% of the tests returned different distances and durations, respectively. The corresponding figures for Hong Kong were 16% and 72%. While many of the differences were minor, we did find notable bad cases. Two such examples are given in Fig. 11.

Fig. 11a shows that, given a starting point (Portcullis House) and a destination (67 Bridge St), Google Maps returned a route with a distance of "3 ft"; but swapping the origin and destination yielded a route of "0.5" miles (Fig. 11b) [8]. From the users' perspective, such a dramatic difference is unacceptable. Worse, as shown in Fig. 11b, "This route has restricted usage or includes private roads." As another example, Figs. 11c [9] and 11d [10] show a similar bad case in Hong Kong. We inspected all of the returned routes to ensure that they did not contain any one-way restrictions for pedestrians.

6.3 Implications

We have shown that, using a "change direction" MR (the functionality for which can be readily accessed by simply clicking on the ⬍ icon shown on the screen), users and testers can easily identify bad cases such as those shown in Figs. 11b and 11d. Because these routes entail unreasonably and unnecessarily longer distances and durations compared with their *symmetric* counterparts (Figs. 11a and 11c), the users should use the routes given by Figs. 11a and 11c, regardless of the direction of travel. For example, if the users want to walk from 67 Bridge St to Portcullis House, they should obviously follow the route given by Fig. 11a (3 feet), albeit in a reverse order, instead of the route given by Fig. 11b (0.5 miles).

8. This inconsistency was repeatable for at least half a year from late 2017 to early 2018, when the issue was rectified around April 2018.

9. URL: https://www.google.com/maps/dir/Mei+Foo+Sun+Chuen+Stage+7+-+1-3+Mount+Sterling+Mall,+1-3+Mount+Sterling+Mall,+Hong+Kong/22.3370167,114.1412827/@22.3374341,114.1379989,17z/data=!3m1!4b1!4m9!4m8!1m5!1m1!1s0x3403f8ab6af1cdfb:0x5d48e5358a826c08!2m2!1d114.1393603!2d22.3375777!1m0!3e2

10. URL: https://www.google.com/maps/dir/22.3370167,114.1412827/Mei+Foo+Sun+Chuen+Stage+7+-+1-3+Mount+Sterling+Mall,+Hong+Kong/@22.3385225,114.1375424,17z/data=!3m1!4b1!4m9!4m8!1m0!1m5!1m1!1s0x3403f8ab6af1cdfb:0x5d48e5358a826c08!2m2!1d114.1393603!2d22.3375777!3e2
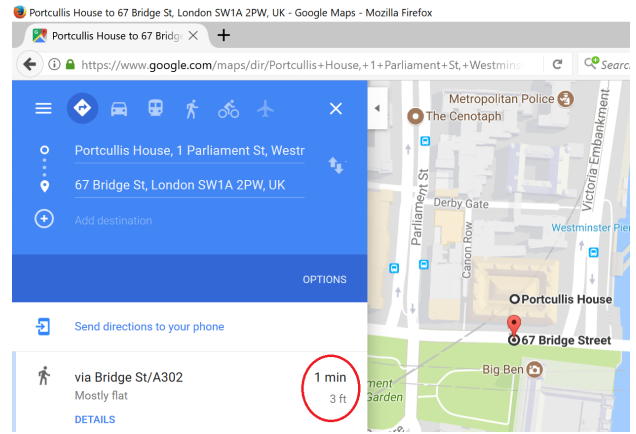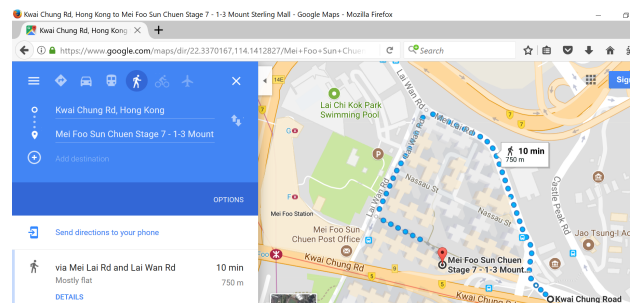

(a) Walking navigation in London: 3 ft, 1 min.
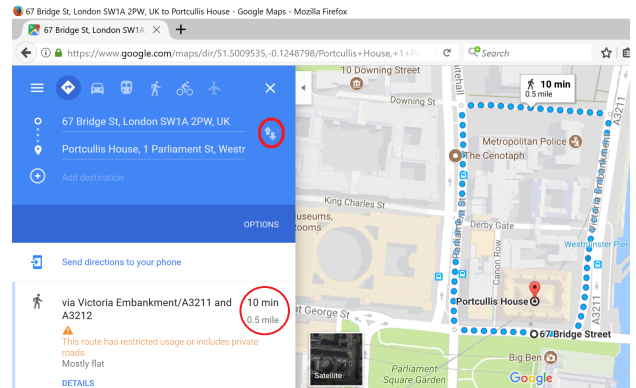
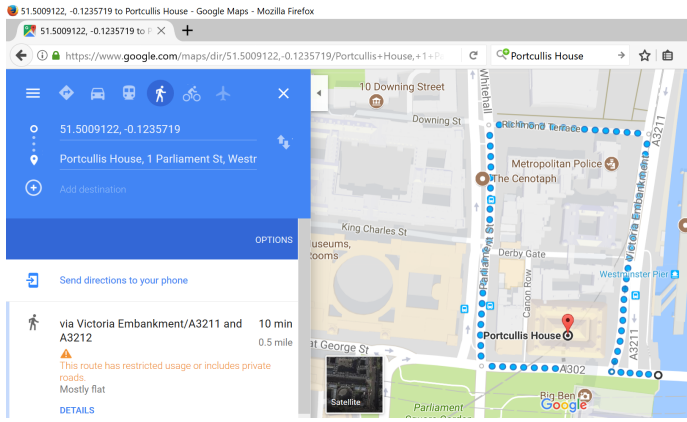(b) Bad case detected: MR violation after reversing origin and destination: 0.5 miles, 10 min.
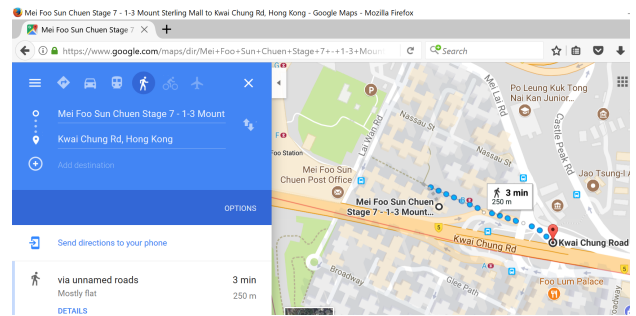
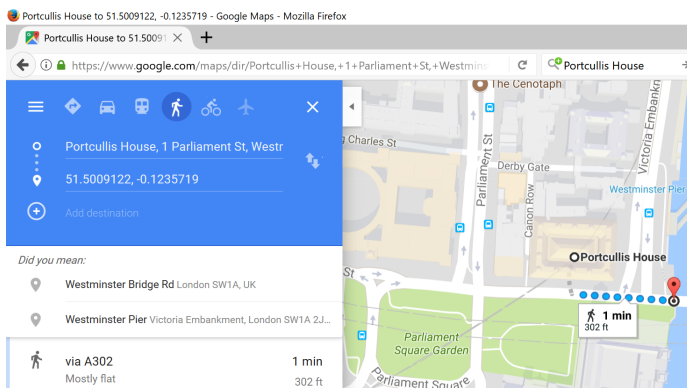(c) Walking navigation in Hong Kong: 250 m, 3 min.

(d) Bad case detected: MR violation after reversing origin and destination: 750 m, 10 min.

Fig. 11: A "change direction" MR revealed Google Maps navigation defects in London (a&b) and Hong Kong (c&d).

(a) The same bad case as in Fig. 11b but with a slightly different starting point.

(b) The bad case was detected and the shortcut was found by swapping the origin and destination, using the same "change direction" MR.

Fig. 12: Detecting the same bad case in another route, in London.

It could be argued that, because the origin and the destination are only three feet apart in the bad case shown in Fig. 11b, it should be so obvious to a user that comparing with Fig. 11a should be unnecessary. There are, however, two points that should be noted: First, navigation software is increasingly used not only by humans, but also by autonomous machinery, such as delivery robots [67], which do not have the common sense of humans, and will strictly follow the directions given by the control systems. Our MR, if adopted by the control systems, would enable the robots to find an alternative and better solution for its routing tasks, without the need for changing the underlying navigation software. Second, bad cases similar to that of Fig. 11b can appear in other routes, such as the one shown in Fig. 12a, where we slightly moved the starting point to be a little farther away from the destination point (and hence the users might not be able to so easily realize the existence of a shorter route). Fig. 12b shows that the "change direction" MR can help users to identify the existence of the shorter route, which is a better solution.

7 CASE STUDY OF LOCATION-BASED SEARCH

In the definition of the symmetry MRP (Definition 2), "the system appears the same" does not mean that the system's source and follow-up outputs must be equivalent. For example, consider the following symmetry of the physical world: In an open space, all other things being equal, if person A can see person B then person B should also be able to see person A. To map this symmetry to the virtual world, we can have the following MR:

$MR_{SeeEachOther}$: In some software systems, users can reasonably expect that if they can find B from A then they can also find A from B, where A and B denote two entities in the system.

$MR_{SeeEachOther}$ is an instance of both the symmetry MRP and the "change direction" MRIP—the "direction" element is the direction "look from A to B" or "look from B to A." In $MR_{SeeEachOther}$, the source output is "B" and the follow-up output is "A"—these two outputs are not equivalent, but they provide a perspective from which the users can understand that the system works in the same way.
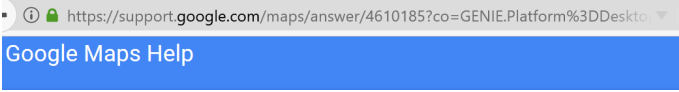
Users can expect many computer systems to demonstrate the symmetry property $MR_{SeeEachOther}$. For example, a typical student management system allows users to find all the students of a given professor; it also allows the users to find all the professors of a given student. Here, A is the professor and B is the student.

In this section, we examine whether or not $MR_{SeeEachOther}$ is satisfied in a location-based search. To do this, we conducted a case study using the "search and find nearby places" feature of Google Maps (see Fig. 13). Failures to satisfy $MR_{SeeEachOther}$ do not necessarily imply a software fault. **However, they do allow the users to better understand how the system works and, hence, to use the system in a smarter way to better meet their information needs.**

7.1 Searching for Cafes in London and Hong Kong using Google Maps

The number of location-based online searches has grown dramatically in recent years. It has been reported that one in five searches is now location-related [68]. According to Google, location targeting helps businesses to focus their advertising on the areas where they will find the right customers, and restrict it in areas where they will not [69].

In this series of experiments, we followed the procedure given in Fig. 13, which was taken from the Google Maps online help page. First, we set the *current location* to a Hilton hotel in a city (when there were multiple Hilton hotels, we randomly chose one). Then, from the *current location*, we searched for "nearby" cafes, using [cafe] as the query term. Let $A = \{a_1, a_2, \ldots, a_n\}$ denote the set of search results (cafes near the Hilton

Fig. 13: Search and find nearby places: A screenshot taken from the Google Maps online help page.

hotel)[11]. Next, for each $a_i \in A$, we set the *current location* to a_i and then searched for "nearby" cafes again. Let $B_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,n_i}\}$ denote the set of these search results, that is, the set of cafes near a_i. As a result, we obtained the following pairs of cafes for each subject city:
$(a_1, b_{1,1}), (a_1, b_{1,2}), \ldots, (a_1, b_{1,n_1}),$
$(a_2, b_{2,1}), (a_2, b_{2,2}), \ldots, (a_2, b_{2,n_2}),$
$\cdots,$
$(a_n, b_{n,1}), (a_n, b_{n,2}), \ldots, (a_n, b_{n,n_n}).$

We selected 100 pairs of cafes for London and another 100 pairs for Hong Kong. For each pair of cafes (a, b), we set the *current location* to b and then searched for "nearby" cafes to see whether a was included in the results list. If the answer was yes, then a and b could find each other; if the answer was no, then a could find b, but b could not find a, that is, $MR_{SeeEachOther}$ was violated. We found that the MR violation rates for London and Hong Kong were 27% and 5%, respectively. These figures mean that the location-based search service offered by Google Maps was not really symmetric. For example, when the *current location* was at St. James Cafe (41 Pall Mall, London) we were able to find a "nearby" cafe named Rainforest Cafe (20 Shaftesbury Ave, London); however, at the latter location we were unable to find St. James Cafe.

We next investigate how our findings can help users with their information needs, by making use of the *asymmetry* of the system.
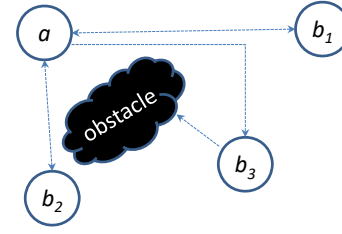
Fig. 14: Asymmetry in the Google Maps location-based search: (*i*) a can see b_3, but b_3 cannot see a; (*ii*) the "obstacle" only blocks the view of b_3 but not the views of b_1 and b_2—b_1 and b_2 can be very close to b_3 but, for some unknown reason, their views are not blocked.

7.2   Making Use of Asymmetry

A high violation rate of the symmetry property $MR_{SeeEachOther}$ indicates that the Google Maps location-based search is *asymmetric*: When a considers b to be a nearby entity, b may or may not consider a to be a nearby entity; in other words, "nearby" is not solely decided by distance.

This situation is depicted in Fig. 14: b_1, b_2, and b_3 are close to a; a can see b_1, b_2, and b_3; b_1 and b_2 can also see a; however, for some unknown reason (represented by an "obstacle"), b_3 cannot see a. We do not know the exact nature of the "obstacle"—it could be a design decision in the Google Maps search algorithm, or an error in the online data/tags, or a fault in the implementation—from the users' perspective, we simply call it an obstacle.

Now, the question is: Can we help the users, whose *current location* is at b_3, to see a? By referring to Fig. 14, we can design two solutions that can be readily adopted. Both solutions make use of the *asymmetry* of the system: The first solution is to slightly change the *current location*; and the second solution is to get rid of the obstacle using a concept of "parallel universes."

7.2.1   Changing the Current Location

In the physical world, when two people stand next to each other, their views should be similar. That is, they see similar things, so the world appears the same to both of them. [12]

Our findings revealed that the above physical rule does not hold in the virtual world of Google Maps. In Fig. 14, the physical distance between b_1 and b_3, or between b_2 and b_3, could be very small. (It should be noted that the "is close to" relation is not transitive. Therefore, in theory, the physical distance between b_1 and b_3, or between b_2 and b_3, may or may not be "very small." However, in practice, it is very possible for some of b_1, b_2, and b_3 to be close to each other. In our experiments, we found many

---

11. Google Maps can return "nearby" places that are located far away in different countries. In all experiments of location-based search, we inspected the search results to exclude such extreme cases from consideration.

12. This symmetry is a result of the properties of light propagation, such as the laws of rectilinear propagation (the property of light travelling in a straight line in a homogeneous transparent medium), reflection, refraction, and independent propagation [70], [71].

cases where the distance was indeed very small. Our current discussion is with regard to this kind of situation.) It is unclear why the obstacle only blocks the view of b_3, and not the views of b_1 and b_2. This observation means that, somehow, the "light" (information flow) in the virtual world does not travel in a "straight line," and that the virtual world is not "homogenous" (it is nonuniform in structure or composition). In short, the system is asymmetric.

The users can make use of this type of asymmetry as follows: They could change their *current location* (slightly) from b_3, for example, by moving away from b_3 and towards b_1 or b_2. It should then be possible for users to see a at certain points, even though the physical distance between the users' new *current location* and a may not decrease. In the worst case, when the users reach b_1 or b_2, they should see a.

The users, of course, are probably unaware of the existence of a, and the fact that a can be found from b_1 and b_2. However, the general strategy underlying our method will still work: If a user is not satisfied with the search results returned by Google Maps, or if he/she wants more results, then changing the *current location* may help. Given the non-homogeneity (and possibly non-linearity) of the software system, even a small change in the *current location* may reveal additional search results that had previously been unseen. This can be achieved by (slightly) changing the *current location* in the Google Maps website. Mobile users could also walk or drive for a small distance and then search "nearby" again.

For example, we used Google Maps from Australia and set the *current address* to `206 Beasley St, Farrer ACT 2607, Australia`. We then searched "nearby" with [`HSBC`] as the query term, and Google Maps returned only one result (see Figs. 15a and 15b).

We then changed the *current address* to the next-door neighbor, `204 Beasley St, Farrer ACT 2607, Australia`, and searched "nearby" for [`HSBC`] again. This time, Google Maps returned ten results, as shown in Figs. 16a and 16b. It should be noted that a new result shown in Fig. 16b was actually closer to the *current location* than the previous result. Fig. 17 shows that the *current locations* of Fig. 15 and Fig. 16 were next-door neighbors.

This example shows that users can make use of the asymmetry of the system, by (slightly) moving the *current location* to achieve more results.

7.2.2 Removing the Obstacle through "Parallel Universes"

Our next question is: Can users find a without moving away from b_3 (Fig. 14)? We next present a strategy for achieving this. We start by borrowing the idea of a *multiverse*, a hypothetical set of *parallel universes*. We then explain that, although a multiverse is only a scientific hypothesis, or philosophical speculation, it can, nonetheless, be a reality of the virtual world in computer systems. Then, based on a further series of experiments
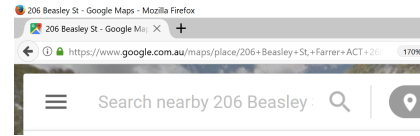
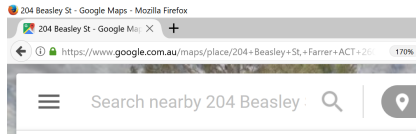(a) The *current location* was set to 206 Beasley St, Farrer ACT 2607, Australia.
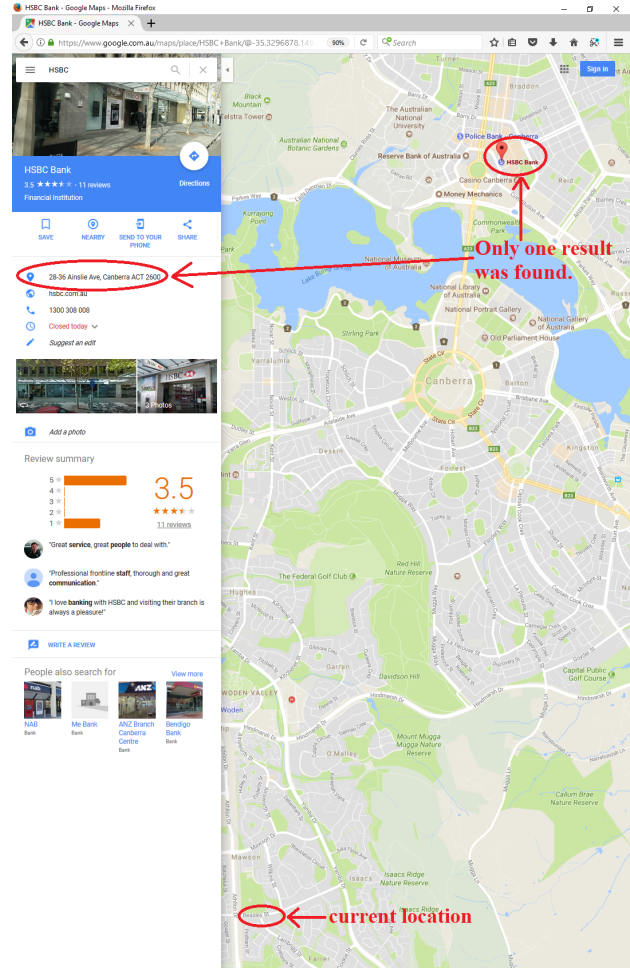
(b) Based on the *current location*, Google Maps found only one result for the query term [HSBC].

Fig. 15: Google Maps found only one "nearby" HSBC when the *current location* was set to 206 Beasley St, Farrer ACT 2607, Australia.

to validate whether the multiverse of the virtual world is symmetric, we show how our findings can help users to remove the "obstacle" shown in Fig. 14 without changing their *current location*.

7.2.2.1 **Multiverse as a scientific hypothesis or philosophical speculation**: A "multiverse" is a hypothesis among cosmologists and theoretical physicists that refers to the idea of parallel universes [72].[13] In science fiction, the concept of a "doppelgänger" (a "double" of

---

13. In spite of the use of the word "parallel," parallel universes should not be assumed to be completely symmetric.

(a) The *current location* was set to 204 Beasley St, Farrer ACT 2607, Australia.
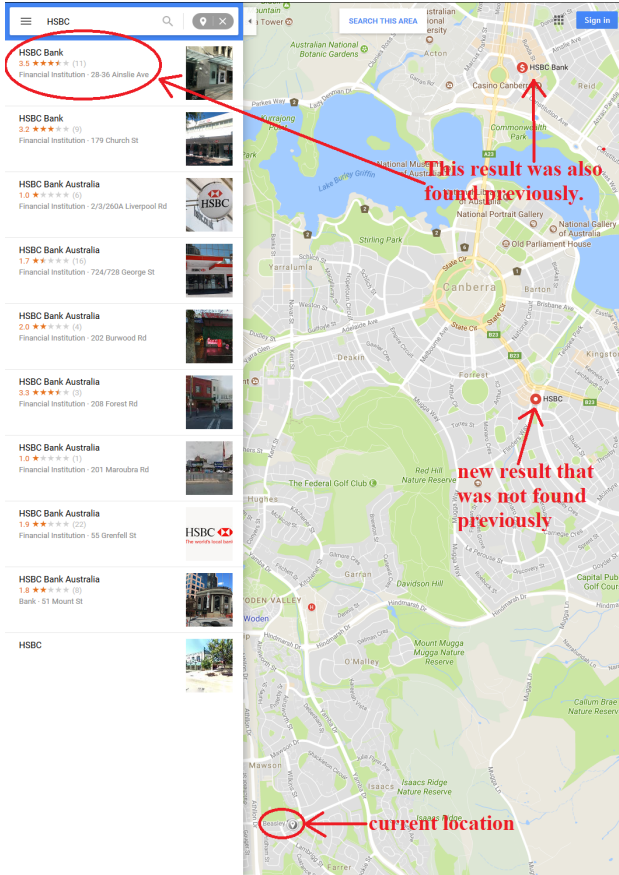
(b) Based on the *current location*, Google Maps found ten results for the query term [HSBC]. Two results are shown in the figure, where the new result was closer to the *current location* than the previous result.

Fig. 16: Google Maps found ten "nearby" HSBC when the *current location* was set to 204 Beasley St, Farrer ACT 2607, Australia.
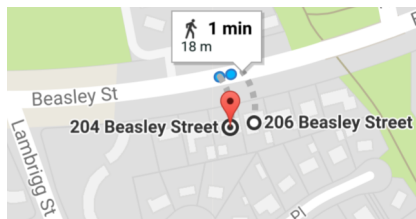

Fig. 17: The *current location* of Fig. 16 was next-door to the *current location* of Fig. 15.

a person) is a frequent theme, with stories varying in how similar the doppelgängers are to their originals [73]. Related to this is *Schrödinger's cat*, or the *cat paradox*,

a thought experiment devised by Erwin Schrödinger, Nobel laureate in Physics, to illustrate what he saw as the problem of the Copenhagen interpretation of quantum mechanics. This involves an imaginary cat in a box, with the cat being simultaneously both alive and dead (until the state has been observed) as a result of being linked to a subatomic event that may or may not occur [74].

In the scientific community, the issue of whether the multiverse is a scientific hypothesis, or just philosophical speculation disguised as science, continues [72]. A comment on the NASA website [14] is intriguing: "Do nearly exact copies of you exist in other universes? If one or more of the multiverse hypotheses is correct, then quite possibly they do." The website also points out that "one criticism of multiverse hypotheses is that they are frequently difficult to test. Some multiverse hypotheses may therefore be great fun to think about but not practically falsifiable and therefore have no predictive scientific value."

7.2.2.2 **Multiverse of the virtual world in the context of a distributed system**: In this study, we borrow the terms "multiverse" and "parallel universes" to describe the virtual world defined by a distributed computer system. This is because a large distributed system such as Google consists of thousands of independent servers [15] running in parallel in different countries to process user queries sent from different locations. In a sense, a server is like a "universe" of the virtual world, and hence the entire distributed system is a multiverse consisting of many "parallel universes" (parallel servers) that coexist, and can work quite independently of each other. If we consider a software entity (such as a functional component or data item) to be a person, then many software entities can find their doppelgängers in other parallel universes within the same distributed system because of the replication of data and processing. The parallelization, concurrency, and synchronization mechanisms of the distributed system should ensure consistency among these doppelgängers so that a user will receive the same computation results regardless of which server processes the request (that is, regardless of which doppelgängers the request encounters). Hence, the existence of the "parallel universes" should be *transparent* to users— this is indeed a basic requirement for the design of a distributed system. In the context of distributed systems, *transparency* "deals with hiding the implementation policies from the user" [75, p. 23], including access transparency, location transparency, migration transparency, transaction transparency, failure transparency, replication transparency, and so on. *Replication transparency* is "a distribution transparency which masks the use of a group of mutually behaviorally compatible objects to support an

---

14. https://apod.nasa.gov/apod/ap101114.html

15. Here, our concept of "servers" is not strictly defined, and is used for illustration purposes only. A server does not need to be physical; it is a subsystem such as one or a group of processes that work independently of other servers. A server may involve multiple hardware devices, and a hardware device may also host multiple servers.
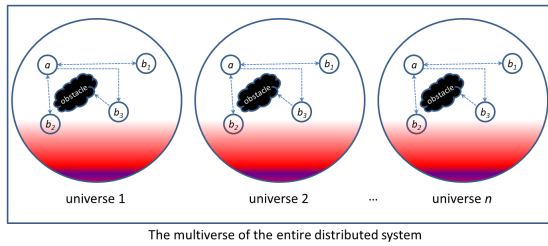
Fig. 18: A highly symmetric multiverse where each parallel universe (universes $1, 2, \ldots, n$) contains exact (equivalent) copies of doppelgängers (a, b_1, b_2, b_3, and the obstacle).
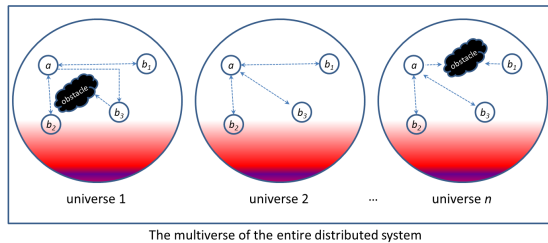

Fig. 19: A less symmetric multiverse where each parallel universe (universes $1, 2, \ldots, n$) may contains different (nonequivalent) copies of doppelgängers (a, b_1, b_2, b_3, and the obstacle).

interface" [76, p. 299] and "does not let the user become aware of any replication" [75, p. 24].

7.2.2.3 **Discussion**: Are the doppelgängers in the parallel universes of the Google Maps multiverse really exact/equivalent copies of each other? If the answer is yes, then these parallel universes must be highly symmetric as illustrated in Fig. 18. This means that if the processing of the user request switches from universe 1 to universe 2, then the user should receive the same results because what happens in universe 1 must also happen in universe 2—for the same person, the different universes tell the same story.

If the answer is no, then these parallel universes are less symmetric, as illustrated in Fig. 19. This means that if the processing of the user request switches from universe 1 to universe 2, the user might receive different results because what happens in universe 1 may not necessarily happen in universe 2—for the same person, the different universes may tell different stories: a doppelgänger may have different states in different universes. For example, the scenario described in Fig. 14 could just be a snapshot of universe 1 depicted in Fig. 19, where there is an obstacle from b_3 to a. However, this obstacle may not necessarily exist in universe 2, and in universe n there could be a different obstacle somewhere else. In a sense, the multiverse depicted in Fig. 19 is like a black box, in which sits Schrödinger's cat (the obstacle), in both alive (as in universe 1) and dead (as in universe 2) states—

the state is probabilistic until the user request has been processed in one of the universes and the results have been observed by the user.

7.2.2.4 **Hypothesis**: In Section 7.2.1, our results revealed a degree of non-homogeneity in the Google Maps system, indicating the system is nonuniform in structure or composition. We therefore hypothesize that the Google Maps multiverse is less symmetric—it is more like Fig. 19 than Fig. 18. If our hypothesis can be proven to be true, then we could get rid of the obstacle by designing a mechanism to move from universe 1 to universe 2 (Fig. 19). From this perspective, our concept of multiverse for the virtual world is different from that for the real world introduced in Section 7.2.2.1, where the parallel universes were not allowed to communicate with each other.

7.2.2.5 **Evaluation of our hypothesis**: We first conducted a small experiment in October 2017, when the first user repeated the query shown in Fig. 16 from Singapore, and the second user repeated the same query from Australia. Queries from both locations were issued and repeated several times during the same period of time. The second user (who queried from Australia) was able to receive the ten results shown in Fig. 16, but the first user (who queried from Singapore) could only receive one result. This observation suggests that the two users' queries were processed by different Google Maps servers, and the different servers returned different results. In other words, the "obstacle" encountered by the first user was not encountered by the second user.

Inspired by this observation, we conducted a further series of experiments to more systematically investigate this phenomenon: From Australia, we issued the same set of queries to two different domains of Google Maps during the same period of time: The first was the default ".au" domain at maps.google.com.au; and the second was the ".com" domain at maps.google.com—we used a US-based VPN service to access this .com domain to ensure that a Google Maps server other than that for .au was being used: It had previously been reported that "Google searches now correspond to user location instead of domain" [68]. In each query, we first set the *current location* to a suburb (town)—in total, we used 58 Australian towns. We then searched for a nearby [HSBC]. The test results showed that the two Google Maps servers returned the same numbers of results for only 25 of the 58 towns, giving an inconsistency rate of 57%. For 15 towns, the .com server returned more results; and for 18 towns, the .au server returned more results.

We have thus proven the hypothesis of Section 7.2.2.4 to be true, which means that Google Maps is like the system depicted in Fig. 19, and that users may be able to remove some of the obstacles by changing the server or platform.

7.2.2.6 **A final example**: In real life, we cannot expect the users to have access to a VPN server located in a different country. Nevertheless, the general idea of changing platforms is still valid, because different

platforms may correspond to different servers[16]. Such an example is shown in Figs. 20, 21, and 22, where all searches were conducted *during the same period of time and from the same location*, and *all search results were repeatable* at the time of the experiment. The user was physically located at the Sheraton Buenos Aires Hotel & Convention Center, and wanted to find a foreign currency exchange. Therefore, using the Google Maps app for Android installed on his Huawei Mate 9 Pro mobile phone, he searched for [casa de cambio] (Spanish for foreign currency exchange). Before he began the search, he disabled all the filters (such as rating scores and opening hours) to maximize the results list (Fig. 20a). He did not need to enter his current location as the Google Maps app automatically obtained his mobile phone's GPS location information and returned a total of *four* results, as shown in (Figs. 20b and 20c).

The user was not satisfied with just four results, and so, using our recommended strategy, he decided to change platform, hoping that the asymmetry of the Google Maps product family would give him more results. Therefore, with the same mobile phone, he opened a browser to visit the Google website and entered his hotel name as the query term, as shown in Fig. 21a. Then, using the browser, he set his hotel as the *current location* and searched for nearby [casa de cambio]. This time, Google Maps returned *six* results (including *two new results that were closer* to the hotel than some of the previous results), as shown in Fig. 21b.

Encouraged by the fact that the Google Maps website, when accessed through the mobile phone browser, gave him more results than the Google Maps mobile app, the user wondered whether he could find more results by changing the platform again. He therefore opened a browser from his *laptop*, and visited the Google Maps website again. For the same query, the website returned *nine* results, including some closer than previous ones. This was three more results than the mobile phone browser, and five more than the mobile app, as shown in Fig. 22. Comparing the results of Figs. 20 and 22, it can be seen that the latter included new results that were better than some of the previous ones: Cambio Platinum Sa (shown in Fig. 22), for example, was preferable to House exchange Palermo (shown in Fig. 20) in terms of both distance and rating. In any case, factors such as distance, rating score, or opening hours, should not be a cause for the different search results, because all filters were disabled at the start of the experiment. Because the results were repeatable at the time of the experiment, it can be concluded that the differences were not caused by web dynamics, either.

Through this example, we have shown that the use of asymmetry across different platforms of the same product family can help the user to find more and better

---

16. The users, of course, can also change their application and service provider, for example, by using Microsoft Bing instead of Google. In this research, however, our focus is on the same product family without assuming the availability of other systems.
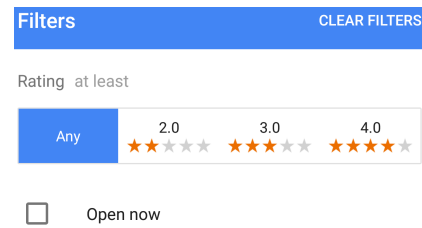
(a) All filters such as rating scores and opening hours were cleared to maximize the number of search results.
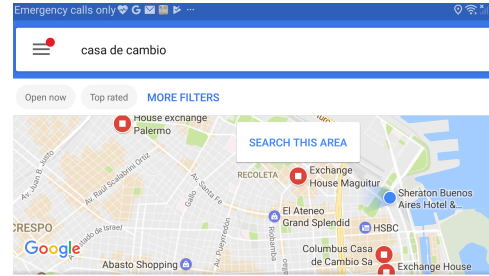

(b) Google Maps mobile app found a total of four results.
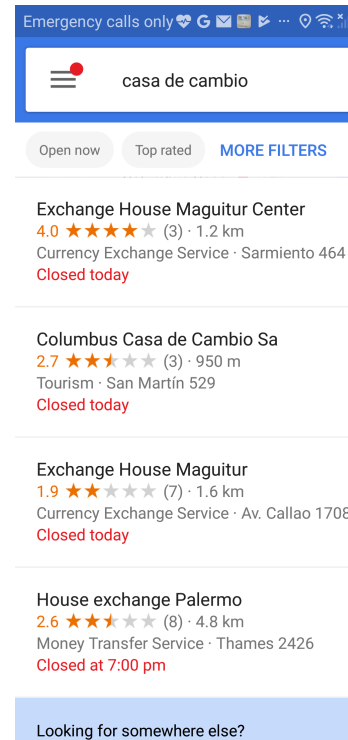

(c) Details of the four results.

Fig. 20: Google Maps mobile app found four results.

results. A limitation of this approach is that some users might not have access to multiple devices. Nevertheless, even in such situations, the user could still use different software from the same device to perform the tasks, such as using a mobile app and a web browser, both from the same mobile phone, as illustrated in Figs. 20 and 21. Furthermore, when two or more people are travelling together, they are likely to have different brands or
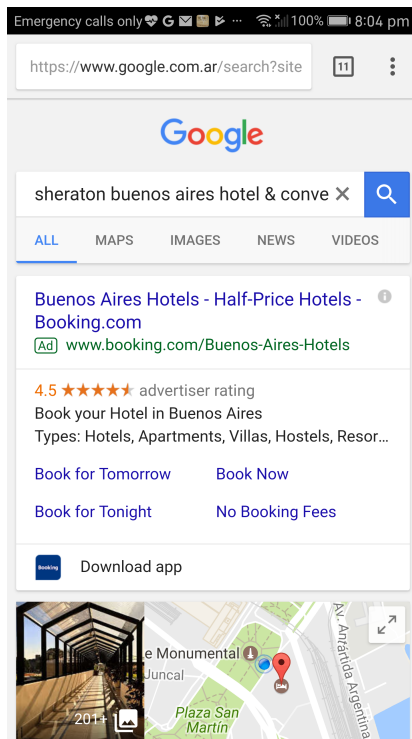
(a) Visiting the Google website using a browser of the mobile phone.
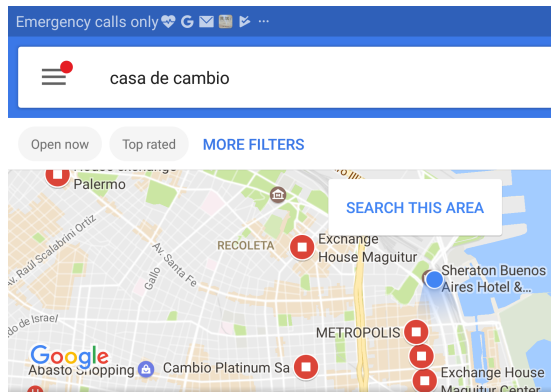
(b) The Google Maps website found a total of six results for the mobile phone user.

Fig. 21: Google Maps website, when accessed from a browser of the mobile phone, found six results.

models of mobile phones, and therefore they can perform the same tasks using their respective phones, and then pool the results.

7.3 Summary

In this section, we have discussed an investigation into Google Maps location-based search. We started by defining $MR_{SeeEachOther}$, an instance of the symmetry MRP and the "change direction" MRIP. In $MR_{SeeEachOther}$, the source and follow-up outputs were not equivalent.

Our experiments first revealed that the system was not really symmetric in terms of $MR_{SeeEachOther}$. We further

showed that this finding has profound implications, and demonstrated that users can make use of this (the asymmetry of the system) to find previously hidden objects or targets. Our first strategy was to set a slightly different *current location*, and the second strategy was to try to force the selection of a different server, by either using a VPN or a different member of the product family. In our next series of experiments, we compared the search results provided by two different Google Maps servers and confirmed that different servers could indeed return very different results. Ordinary users seldom consider changing servers to achieve better search results. Modern distributed system design principles also require that the existence of multiple servers be transparent to users (that is, users should not be aware of their existence), which means that, at least theoretically, different servers should return the same results. All our discussions here have been supported by experimental results and comprehensive, real-life examples and case studies.

8 CASE STUDY OF IMAGE ANALYSIS USING MATLAB, OPENCV, AND FACEBOOK

All of the applications studied so far have involved certain kinds of user queries. In this section, we investigate a different type of software, one that does not involve textual queries from the users: image analysis software for face recognition.

In our MR, we change the direction of the x-axis of the image to create a mirror image, and the symmetry is that a mirror image of a face is still a face, because human faces have approximate bilateral symmetry.

8.1 The Systems Under Test

Matlab is a well-known commercial off-the-shelf package for scientists and engineers. It provides a numerical computing environment with multiple toolboxes. In our experiment, the Computer Vision System Toolbox *cascade object detector* [17] of Matlab version R2017b 64-bit was used to detect faces in photos. Photos were read by the function *imread*, and the function *flip* was used to obtain a mirror image.

OpenCV (Open Source Computer Vision Library) is a popular open source computer vision and machine learning software library. In our experiment, we used OpenCV version 3.3.0 with a Python interface. The function we tested is cv2.CascadeClassifier.detectMultiScale [18] with the pre-trained classifiers configuration file [19] for the front of the face. All parameters used default values. Photos were read by the function *cv2.imread* and were flipped by the function *cv2.flip*.

17. https://au.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html
18. https://docs.opencv.org/3.3.0/d1/de5/classcv_1_1CascadeClassifier.html
19. https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
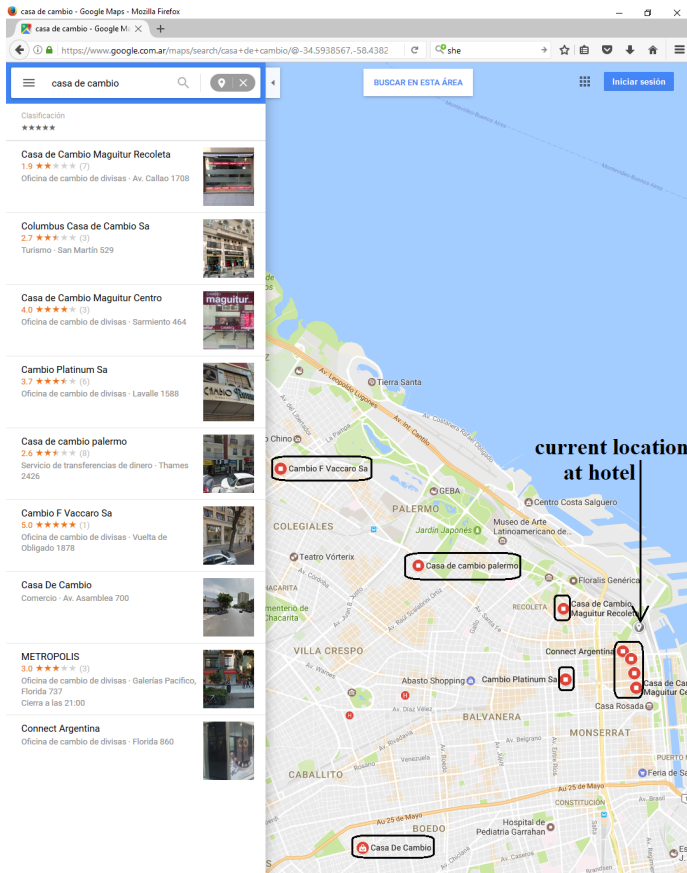
Fig. 22: Google Maps website, when accessed from a browser of a laptop, found nine results.

(a) Share post with photo. Click the "Tag" button to edit photo.

(b) Face being detected in an uploaded image.

Fig. 23: Automatic face detection in https://www.facebook.com.

Facebook (https://www.facebook.com) allows users to upload images as part of a posting (for example, see Fig. 23a). The system can automatically analyze photos and identify faces in them. When a user tags a photo [20], if a face is detected, the system prompts with a hint "Who is this?" and draws a box on the face (for example, see Fig. 23b).
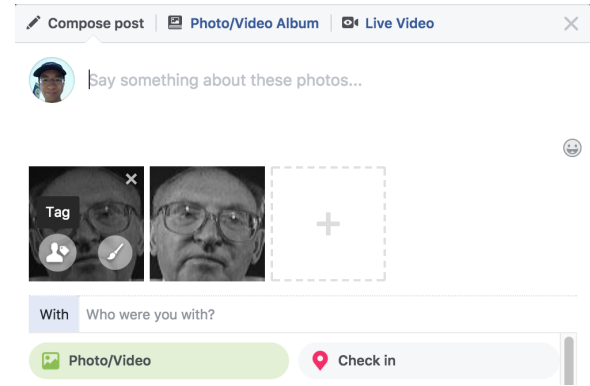
8.2 Test Cases

All photos used in our experiments were taken from the *Database of Faces* of AT&T Laboratories Cambridge, hosted in conjunction with the Cambridge University Computer Laboratory [21]. The database contains ten different images for each of the 40 distinct subjects in an upright, frontal position, giving a total of 400 images.
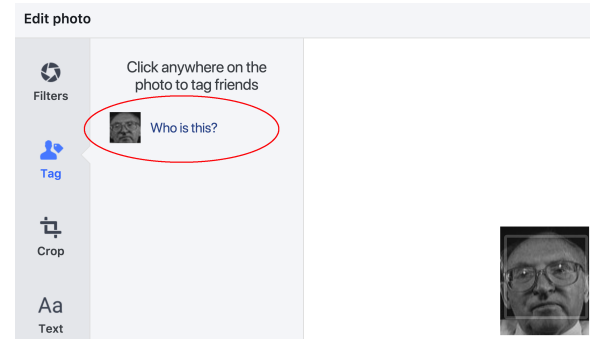
8.3 Results

Table 4 summarizes the test results. Type 1 violations refer to situations where no face was detected in the original photo, but one face was detected in the flipped one. This is the largest category of violations, and no SUTs passed

20. https://www.facebook.com/help/tag-suggestions
21. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

this test. Type 2 violations refer to situations where one face was detected in the original photo, but none were detected in the flipped one. Matlab and OpenCV failed this test. Type 3 violations refer to situations where two faces were detected in the original photo, but only one was detected in the flipped photo. Facebook and OpenCV failed this test. Examples of each violation type for each SUT are given in Fig. 24.

TABLE 4: Results of the face recognition experiments.

SUT	Number of (original, flipped) image pairs	Number of MR violations		
		Type 1	Type 2	Type 3
Facebook	400	1	0	1
Matlab	400	24	20	0
OpenCV	400	14	8	1

8.4 Implications

Our experimental results suggest that users of image analysis software can make use of symmetry properties (including, but not limited to, flipping the image) to detect potential false negative and false positive cases. Developers not only can use this strategy to verify their

(a) Facebook Type 1: original (b) Facebook Type 1: flipped (c) Facebook Type 3: original (d) Facebook Type 3: flipped

(e) Matlab Type 1: original (f) Matlab Type 1: flipped (g) Matlab Type 2: original (h) Matlab Type 2: flipped

(i) OpenCV Type 1: original (j) OpenCV Type 1: flipped (k) OpenCV Type 2: original (l) OpenCV Type 2: flipped

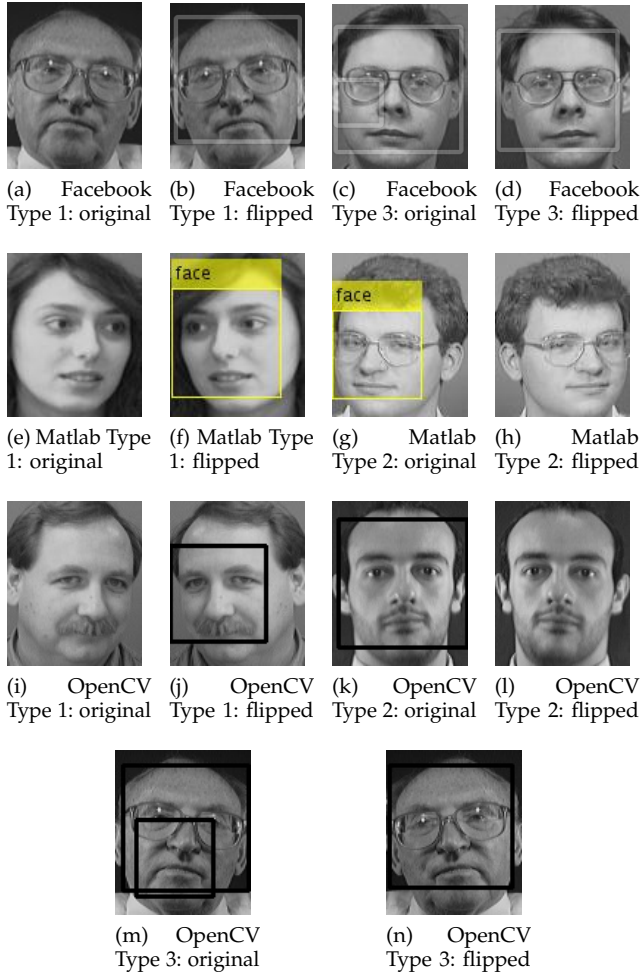(m) OpenCV Type 3: original (n) OpenCV Type 3: flipped

Fig. 24: Examples of MR violations.

software during the testing phase, but also can improve their algorithms or embed some self-checking code in their products by taking advantage of the symmetry properties of images.

9 CASE STUDY OF VIDEO ANALYSIS USING GOOGLE CLOUD VIDEO INTELLIGENCE: EXPLORING THE TIME REVERSAL SYMMETRY

In this section, we report on a small case study investigating video analysis software. The SUT was a service provided by Google Cloud Video Intelligence [77] that can automatically recognize objects inside videos. This functionality makes videos searchable by extracting metadata from the video files and annotating "key nouns entities." In this case study, we used a Google API demo page [78] to conduct the testing. The video analysis service took a user uploaded video as input, listing the detected objects together with the associated confidence scores.

9.1 The Test Case and the MR

We used only one test case, a 39-second video that only included five static objects (a cup, a keyboard, a knife, a

hair dryer, and a headphone), as shown in Fig. 25. We recorded the video using a mobile phone.

Using the symmetry MRP and the "change direction" MRIP, we identified the following MR: The SUT should find the same set of objects in the video regardless of whether it is played forwards or backwards.

To create a follow-up test case, we used a tool *ffmpeg* (version 3.1.1, https://www.ffmpeg.org) to reverse the original video by playing it backwards.

9.2 Results

In the original video, the Google Cloud Video Intelligence API detected four objects (Fig. 26a), but in the backwards-played video, five objects were detected (Fig. 26b). In Figs. 26a and 26b, the labels on the left are the names of the recognized objects, and the percentages on the right indicate the associated confidence. It should be noted that a knife was detected in Fig. 26b with a high degree of confidence (72%), but not detected at all in the original video (Fig. 26a). We repeated the experiment several times using the sequence: original video, reversed video, original video, reversed video. The same results were obtained each time.
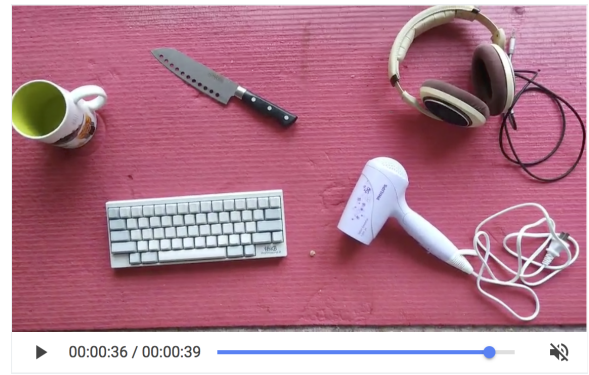

Fig. 25: A snapshot of the video (our test case).

9.3 Implications

The results reported in this section have similar implications to those discussed in Section 8.4. Although the case study was of a very small scale, it nonetheless shows a significant application of the *time reversal symmetry* (*T-symmetry*).

10 DISCUSSION

In this section, we first clarify the relationship between the present work and MT, and then discuss the validity and sufficiency of MRs. We then examine the design space and choices in our methodology. We conclude this section by analyzing customers' roles in related software processes.

Video Labels

Detect objects, such as dog, flower, human, in the entire video.

headphones   84%

audio equipment   53%

audio   46%

computer keyboard   43%

Note: *shot-level label annotations are present on the shots tab.*

(a) Four objects were detected in the original video.

Video Labels

Detect objects, such as dog, flower, human, in the entire video.

headphones   84%

knife   72%

audio equipment   53%

audio   47%

computer keyboard   44%

Note: *shot-level label annotations are present on the shots tab.*

(b) Five objects were detected in the backwards-played video.

Fig. 26: Test results of Google Cloud Video Intelligence.

10.1 Metamorphic Testing and Metamorphic Exploration

Metamorphic testing was originally proposed as a verification technique, where MRs were identified based on the software specification or the target algorithm [23], [24]. Recently, Zhou et al. [9] extended MT into a paradigm that covers not only verification but also validation and other types of software quality assessment, where MRs can be identified not only based on software specifications, but also based on user or stakeholder expectations. Generally speaking, in the MT literature, if an MR is violated for certain test cases, then the software is considered to be faulty, or at least to have failed to meet user expectations [26], [28].

In this paper, although we have also shown the detection of issues in the SUTs (some of which revealed real-life, previously unknown bugs), the main focus of this work is not on the detection of bugs, but rather on what users can do when faced with imperfect or unfamiliar software. For this purpose, the MRs are used as a tool for users to explore the system to better understand how it works. Accordingly, our method is also different from software fault tolerance and self-healing techniques such as *data diversity* [79] and *automatic workarounds*

[80]. Here, an MR violation may not necessarily imply a software fault or user dissatisfaction, but instead allows the user to better understand the system and, if necessary, to take actions to achieve more desirable computation results (such as by using a follow-up input for a certain MR). Strictly speaking, the user is not performing software testing for fault detection or quality assessment, but is instead using the MRs to explore the system and to better satisfy his/her specific information needs. We therefore call this ***metamorphic exploration***, which is similar to MT, but is performed by users for a different purpose: Enhancing system understanding so that the system can be used to serve them better.

10.2 Validity and Sufficiency of Metamorphic Relations

This section answers the following three questions: (1) Do MRs need to be validated? (2) Could the validation of MRs also suffer from the oracle problem? (3) How many MRs need to be identified/used?

In the context of metamorphic testing, MRs are expected properties of the SUT. The tester, therefore, needs to ensure that the MRs are valid. So, the answer to question (1) is yes. Furthermore, in some situations, it may not be easy to prove that the MR under consideration is indeed a necessary property of the specified functionality of the software. Therefore, the answer to question (2) is also yes. However, because an MR is only an expected property of the software (that is, one of possibly many expected aspects of the software), validating the MR should be easier than validating the entire software. For example, it is straightforward to confirm that $\sin \theta = \sin (\theta + 2\pi)$ is a valid MR (which can be done by referring to a mathematical textbook), but is not easy to verify the output of a program implementing the sine function. To address question (3), Liu et al. conducted an in-depth study [20], and showed that, for the purpose of software verification, a small number of (on average, around six) diverse MRs were sufficient to match the fault-detection effectiveness of a test oracle for software implementing a knapsack algorithm.

In the context of metamorphic exploration, MRs are identified by the users based on what they are most interested in, and these MRs *may or may not* be necessary properties of the target system. In this situation, with regard to questions (1) and (2), there is no "valid" or "invalid" MR (because users can try anything they like). In other words, there is no validation or oracle problem for the MRs, as they only represent something that the users want to explore—violating an MR in the actual execution may help them to reflect on their original assumption and thereby gain better understanding of the system; satisfying an MR in the actual execution may help confirm an assumption, and thereby also lead to better understanding of the system. Furthermore, different users may have completely different or even conflicting MRs, all of which can be helpful for exploring

the system. Consider the "American inventors" example given in Section 3.1: In contrast to the student, another user could assume a different MR, such as "the search engine should return different results for the English and Chinese queries, because Chinese searchers might be interested in certain American inventors more than English searchers." This MR contrasts with the one identified by the student in Section 3.1, and may involve a more complex symmetric relation. This user may be happy to find that the Google search results confirm his/her assumption in the sense that queries in different languages do indeed return different results.

In the context of metamorphic exploration, software validation, and quality assessment, MRs are identified to cover the software features or quality characteristics that interest the users or stakeholders. Even for the same system, different users or stakeholders may be interested in different aspects of the system and, therefore, may want to select different MRs for experimentation. For example, those who are not interested in prices will not come up with $MR_{PriceSort}$, presented in Section 5.1. Therefore, for metamorphic exploration, software validation, and quality assessment, the answer to question (3) depends on the interests of individual users and stakeholders.

10.3 The Design of Metamorphic Relations and Comparisons of Choices

In this section, we discuss the design space that led us to the selection of the symmetry MRP, and analyze the relationships between different choices.

10.3.1 Why Symmetry was Selected?

In this study, we raised three research questions. To address the research questions RQ1 and RQ2, we need to design a solution that can help users to better understand and use the system in the absence of a comprehensive system specification and oracle. Therefore, because they are widely recognized in the research community as a highly effective approach to addressing the oracle problem, we decided to make use of MRs. Our third research question, RQ3, requires that our solutions be "simple, applicable, and effective" for users in multiple domains. This means that we need to find a rule (in the context of metamorphic exploration, this rule is expressed as an MR at a higher level of abstraction, that is, a "metamorphic relation pattern") that not only can be widely observed in different areas, but also achieves a good combination of simplicity and strength. We found symmetry to be the best candidate that satisfies all these requirements. For example, in the literature of physics and philosophy, the notion of simplicity is often associated with symmetry [46]. Furthermore, as elaborated in Section 4.4, symmetry is ubiquitous in the real world. Besides, symmetry is also elegant [46], and can be easily understood by ordinary users without a mathematical background. In addition to these attributes, our reflection (see, for example, Section 4.2) also shows

that symmetry is effective. Indeed, as pointed out by Hargittai and Hargittai, "*symmetry is beautiful although alone it may not be enough for beauty, and absolute perfection may even be irritating. Function, utility, and aesthetic appeal are the reasons for symmetry in technology and the arts*" [47, p. 1].

10.3.2 The Design Space and Choices

In the existing literature, the concepts of metamorphic relations and their patterns have been developed for software testing, verification, and validation purposes [9], [26], [28], [37]–[39]; in other words, the existing concepts are not developed to facilitate the objectives specified in our research questions RQ1, RQ2, and RQ3.

In contrast to existing work, in this research we considered how to identify suitable MRs for the purpose of metamorphic exploration, namely, to help users to better understand and use the system. In a sense, to understand a software system by observing its input and output is similar to trying to know a person's appearance by only looking at pictures of them: A front-view photo shows the person's eyes, nose, and mouth; but it does not show full information about the height of the nose, and some other facial features. To know this person's appearance better, we would need profile (side-view) photos, which show the height of the nose as well as some other facial features, but at the expense of losing some front-view information such as the width of the mouth. If the side-view and front-view perspectives only differ by a small angle, they will look similar and still lack information about the height of the nose. In other words, the side-view picture of the same person should be taken from a *different* perspective so that we can know more about the person's appearance [81].

Like taking pictures of the same person from different angles, the symmetry MRP provides "different viewpoints" of the same system. Therefore, we believe that symmetry is a suitable MRP for the purpose of metamorphic exploration to help users to understand the system. In the definition of the symmetry MRP (Definition 2), the key phrase "different viewpoints" corresponds to the variations of input values to the system. When there are many different viewpoints, which one(s) can best serve the purpose? Still considering the front- and side-view photos, as explained, it is not very useful if they deviate by only a small angle. Obviously, a *good* side-view photo to capture the information of the height of the nose is from an angle set 90 degrees apart from the front view [82, p. 226]. Likewise, we need to study metamorphic relation *input* patterns (MRIPs), which describe generic ways of changing input values for the purpose of constructing *good* MRs for metamorphic exploration.

As explained in Section 4.1, previous studies on abstract forms of MRs [37]–[39] were not focused on input. Nevertheless, in the context of automated test case generation, there is a related technique called *data mutation* [83]: Given a set of test cases (seeds), data mutation

generates new test cases by modifying the seeds using a set of operators, called *data mutation operators*. Data mutation operators can include, for example: increase or decrease the value of a parameter; set a parameter to 0, to a very large value, or to its negative; swap the values of some parameters; etc. In the extreme case, the modification to the values can be done randomly, which is fuzzing (fuzz testing). Data mutation is different from metamorphic testing as it only considers ways of changing the values of input parameters without looking at the output relations. Technically speaking, metamorphic relations involve *semantics* rather than data mutation at the syntax level. Nevertheless, data mutation and metamorphic testing can be combined, and have produced practical tools [84]. Data mutation operators can provide heuristics for the identification of potential MRIPs, which will be a focus for our future research.

If the users are smart enough, even without a "pattern," they might still be able to identify suitable MRs to conduct metamorphic exploration. However, the vast majority of users need some guidance. Our concepts of MRPs and MRIPs (and the specific patterns symmetry and "change direction") aim to help such users to narrow down the search space, focus on the most potentially useful properties and, hence, more easily identify useful MRs. Compared with data mutation operators, our concepts of symmetry and "change direction" are at a much higher level of abstraction, and are simple and general enough that they can be easily applied by users to a variety of application domains.

As noted under Definitions 1 and 3, the identification of MRPs and MRIPs can be achieved through levels of abstraction, which immediately suggests two approaches: top-down, and bottom-up. In the top-down approach, a pattern at a high level of abstraction can be first identified and used to derive other patterns at lower levels in the hierarchy, or concrete MRs at the bottom level. In the bottom-up approach, concrete MRs can be first identified by directly analyzing the input parameters of the SUT. Later, multiple concrete MRs can be analyzed to identify commonalities, leading to generalization of some MRs into patterns. The top-down and bottom-up approaches can also be combined.

In addition to top-down and bottom-up, two other general approaches for designing MRs are: input-driven and output-driven. In this research, following the "change direction" MRIP, we only considered the input-driven approach (that is, identification of MRs guided by considering how to modify the input). In contrast to this, the process could also be output-driven, in which the identification of MRs is guided by first considering the output relations—for example, by referring to some metamorphic relation output patterns such as those designed by Segura et al. [39], as explained in Section 4.1. Once an output relation is identified, the kinds of changes to be made to the input in order to achieve that output relation can be considered. For example, in the context of using a web search engine, the users may be interested in the "equality" output relation. Then, in order to make two search results pages equal, the users may think that searching for "A AND B" and searching for "B AND A" should produce the same results, because the logical conjunction of the search criteria A and B is equivalent to that of B and A. When using this relation to conduct metamorphic exploration, users will probably find that the search results, although similar, are not actually identical [9]. A further investigation of this issue would help them to understand that a web search engine works differently from a traditional database application: The MR holds true for the latter but not for the former, because web search engines normally give more weighting to the first query term than to the second [9].

It should be noted that, while in many studies an MR can be decomposed into two separate components— one subrelation involving only the inputs and the other involving only the outputs—this is not the case for all MRs [26]. For example, in Section 6, the MR "swapping the origin and destination should return a route with a similar cost" can be split into an input-only subrelation (where the follow-up input is dependent only on the source input) and an output-only subrelation (where we need to compare the follow-up output only with the source output). In some situations, however, the generation of the follow-up input may also involve the source output, and the verification of the outputs may also involve some of the inputs. Such an example was given in Section 7.1: Let a denote a source input, which is a cafe selected using certain techniques. Based on a, we searched for "nearby" cafes. Let b be one of such cafes returned by the location-based search engine, that is, b is the source output (or part of the source output). In the follow-up search, we set the *current location* to b and then searched again for "nearby" cafes. This means that the source output b has now become the follow-up input. We then checked whether a could be located, that is, whether or not the source input appeared in the follow-up output. This is an example that shows a situation where the follow-up input is based on the source output, and the checking of the follow-up output involves the source input. **In this situation, the MR cannot separate into input-only and output-only subrelations. Nevertheless, our "change direction" MRIP can still be applied in this situation.**

Finally, it should be noted that, although the selection of MRs can be quite different for various purposes, it is always associated with requirements. Therefore, requirements elicitation and analysis techniques can also be applied for the purpose of MR identification.

10.3.3 Symmetry and Asymmetry

In the natural world, symmetry and asymmetry are often observed together. For example, while human bodies appear approximately symmetric, the internal organs are not. As the ancient Chinese philosopher Lao Tzu wrote: "*It is because everyone under Heaven recognizes beauty as*

beauty, that the idea of ugliness exists. ⋯ For truly Being and Not-being grow out of one another" [85].

Similarly, we hypothesize that symmetry and asymmetry are two fundamental MR patterns that come in pairs for computer systems. Furthermore, while this research has been focused on symmetry, we have also shown the applications of asymmetry in our methodology: In the reported case studies, we often first defined a symmetry MR and then showed that the SUT violated it, indicating that the system was, from some perspective, asymmetric. We then showed how users could make use of this asymmetry to achieve more desirable results (as in the example shown in Fig. 19).

We wish to reiterate that while the concept of symmetry is associated with invariance, it does not necessarily imply an equality or equivalence relation (for example, see $MR_{SeeEachOther}$). This is different from *data diversity* [79] and *automatic workarounds* [80], both of which rely on equivalent executions. Furthermore, in a metamorphic relation, the generation of the follow-up input may involve not only the source input but also the source output (see, for example, [26, p. 4:7] and [86, pp. 49-56]), but this is not the case in data diversity [79].

10.4 Customers' Roles

The advantage of early involvement of users in software development projects has been widely recognized [87]. Fig. 27 shows typical verification and validation activities in a conventional software development project [9], [87], where the white arrows correspond to verification activities, and the shaded arrows (a, b, …, f) correspond to validation activities. In the validation activities, various software artifacts are reviewed or tested by the actual users against their real needs.

In the activities depicted in Fig. 27, metamorphic exploration has the potential to help the users to better understand the system or subsystem in the validation activities a, f, and e (where a represents user acceptance testing, and f and e represent user testing activities at the system and subsystem levels).
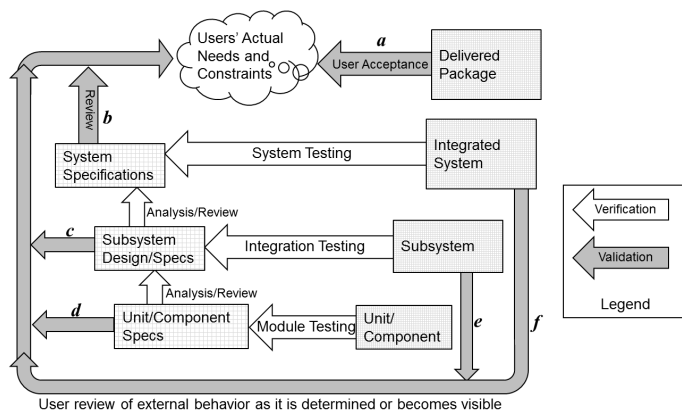

Fig. 27: Conventional software verification and validation activities (taken from [9], which was adapted from [87]).

While developers write tests "method-by-method," customers write tests "story-by-story" [88]. In requirements engineering, a *user story* is a short description of a feature, or software functionality, told from the perspective of the software's user. A user story differs from a *requirement* in that the former normally does not contain implementation information, but instead is focused on user experience. For example, a user story could be told as: "As a home buyer, I want to search for recently sold homes, so I can estimate prices in my area" [89]. We found it straightforward to turn the MRs introduced in the various case studies of the present paper (Sections 5 to 9) into user stories. Therefore, our method is not only useful in the validation stage, but could also be helpful in the requirements stage, guiding users to create their stories using the simple concept of MRs. This shows that our method is also useful for user training in both the requirements and the validation stages.

The effectiveness of our methodology depends on user background and experience, and different users could identify very different MRs. Consider, for example, the image analysis case study of Section 8. Matlab and OpenCV users are generally more technical and, therefore, might be able to design the MR involving the use of mirror images; in contrast, Facebook users are generally less technical and, hence, might not be able to use such an MR. On the one hand, this observation shows a limitation of our approach; on the other hand, it means that the effectiveness of metamorphic exploration can be improved through user training. Examining the amount of training required for users of different backgrounds to be able to effectively use metamorphic exploration will be a future research topic.

Our approach could also be useful in the context of crowdsourced testing. It has been reported that CrowdOracles are a viable solution to mitigate the test oracle problem, but the task of getting useful results from the crowd can be challenging [12], [13]. The simple concepts of metamorphic relations and metamorphic exploration can be introduced to the crowd, so that they can have a way to better understanding the system under test. Furthermore, MRs identified by different individuals naturally contain a degree of diversity (a highly desirable characteristic of effective metamorphic testing, as shown in previous studies, and hence expected to impact on effective metamorphic exploration) [20]. Therefore, as pointed out in our recent survey paper [26], making use of crowdsourcing to brainstorm and identify MRs for a given system is a promising research direction.

11 EVALUATION AND LIMITATIONS

In this section, we validate our approach by first presenting feedback on our findings from the software vendors, who confirmed its value and usefulness mostly from the software developer's perspective. Next, we present a recent news report closely related to our study, which shows its value and usefulness from the user's

perspective. We then present a critical analysis of the limitations of our approach. Finally, we introduce related research and development progress in industry.

11.1 Feedback from the Owners / Developers of the Software Systems

When we detected a bug, and where there was a way to provide a bug report online or by email, we sent such a report to the owner or vendor of the software system. Furthermore, we also personally contacted some software engineers inside the software companies whose products were tested in this study. The feedback we collected is summarized as follows.

With regard to the issues reported in Section 5, the replies we received either confirmed that there were indeed faults in the system or indicated (either explicitly or implicitly) that there was (hidden) business logic in price sorting not disclosed to end users. For example, Overstock.com confirmed the existence of the previously unknown faults and indicated that they would try to fix the problem, whereas Apartments.com explained that their system "will sometimes narrow your search," without giving further details about the underlying algorithm. Representatives from other eCommerce websites also acknowledged the existence of hidden logic in the sorting algorithms (and the complexity of such algorithms), such as the considerations of click-through rates, add-to-cart rates, and so on; they further indicated that the correctness of sorting could also be affected by the quality of the catalog data (including data from third-party sellers). All this feedback confirmed that our findings are valuable and useful for both the developers (for the purpose of fault detection) and the users (for the purpose of better understanding and using the system in the presence of hidden business logic).

With regard to the issues in Google Maps Navigation presented in Section 6, we reported the detected bad cases (Fig. 11) to the relevant Google team, who replied that "that is indeed interesting," and that they were further investigating these problems. Regardless of the investigation outcomes, Google's reply clearly indicated that our approach is useful for them.

With regard to the issues in face recognition reported in Section 8, Matlab confirmed that they were able to "reproduce the issue," and explained that it revealed situations where their pre-trained classifiers "are not always sufficient for a particular application," and that, to solve this problem, users could use another function to train a custom classifier. Matlab's reply means that our approach is indeed useful for users to quickly recognize whether the software is really appropriate for their particular application (as stated in RQ1). Furthermore, we have also received a reply from Facebook, in which they indicated that they were working to fix the technical problems, and that our report was helpful for them to improve the Facebook user experience.

With regard to the issues in video analysis reported in Section 9, Google informed us that the issues were being investigated by their Cloud Video Intelligence API team, but that there was not yet any estimation of when the fix would be available. Google's reply indicates that our findings have interested their development team, and that they are trying to address the problem.

11.2 A Recent News Report Confirming the Usefulness of our Approach from the Users' Perspective

In Section 7, we presented a "multiverse" concept for metamorphic exploration. While interesting, it may be argued that the concept is not practical for a normal user: To be able to derive tests to enhance their knowledge, the user has to learn about both the software product and our metamorphic exploration techniques. In this section, we present a strong case, recently reported in a newspaper, to show a situation where users are very well motivated to apply our techniques to explore the software that they are using.

We submitted the first version of the present paper to the *IEEE Transactions on Software Engineering* on Feb 7, 2018, where in Section 7.2.2.6 we presented an example (Figs. 20, 21, and 22) to illustrate our finding that a mobile application and a web browser could return very different results for the same search query, and that the use of asymmetry across different platforms of the same product family can help the user to find more and better results.

On Feb 19, 2018 (12 days after the submission of our manuscript), the Ming Pao newspaper (明报, a primary Chinese newspaper in Hong Kong) had its *headline* news titled (in English translation): "Booking hotel rooms online using an App and a computer can have 24% price differences from the same travel agent." An excerpt of the original news is shown in Fig. 28a, and its English translation, by Google Translate, is shown in Fig. 28b. In this news report, the tester essentially applied the same approach, as introduced in our Section 7.2.2.6, to investigate hotel room booking prices offered by major online booking businesses including Agoda, Expedia, and Ctrip, and surprisingly discovered a secret that the same online booking business could give very different prices when accessed using different devices. On the one hand, this news report was truly serendipitous as we had not shared our method with the newspaper; on the other hand, reporting this story as the headline news of the day is evidence that their (and hence our) approach is valid and highly relevant to consumers, independent testers, the news media, and even the relevant regulatory authorities. It is also evidence that the various stakeholders should be motivated to learn and use this approach.

11.3 Limitations of our Approach

In this section, we present a critical analysis of the limitations of our approach. At a fundamental level, our approach, metamorphic exploration, is only a partial analysis method, and is dependent on metamorphic relations and test cases. Therefore, as a methodology,

網上訂酒店 App與電腦不同價 同一旅社房價可相差24%

【明報專訊】愈來愈多人在網上訂酒店和機票，同一間酒店於不同網站有不同價錢是常事，但本報測試Agoda、Expedia及Ctrip三大線上旅行社（Online Travel Agents, 下稱OTA）發現，即使同一OTA，用流動裝置或桌上電腦訂同一間酒店房，都會有不同價錢。其中Agoda在不同裝置顯示的價錢差幅最大，iPad App的房價較桌上電腦的便宜24.7%（見圖）。消委會呼籲各旅遊網站提高價格透明度，令市民容易比較經不同途徑購買的價格。

(a) Excerpts of the original Ming Pao news.

URL: https://news.mingpao.com/pns/dailynews/web_tc/article/20180219/s00001/1518976216906

Booking online with a hotel app and a computer at different prices can be 24% different

[Ming Pao News] More and more people are booking hotels and airline tickets online. It is not uncommon for a hotel to have different prices on different websites. However, this newspaper tested Agoda, Expedia, and Ctrip's three online travel agencies (Online Travel Agents, OTA) found that even with the same OTA, booking a hotel room with a mobile device or a desktop computer would have different prices. Among them, Agoda displayed the largest price difference among different devices, and the iPad App's price was 24.7% cheaper than the desktop computer (see figure). The Consumer Council appealed to tourism websites to increase the price transparency so that the public could easily compare the prices purchased through different channels.

(b) English translation (by Google Translate) of the Ming Pao news.

URL: https://translate.google.com/translate?sl=auto&tl=en&js=y&prev=_t&hl=en&ie=UTF-8&u=
https%3A%2F%2Fnews.mingpao.com%2Fpns%2Fdailynews%2Fweb_tc%2Farticle%2F20180219%2Fs00001%2F1518976216906&edit-text=

Fig. 28: Ming Pao headline news, Hong Kong, Feb 19, 2018 (12 days after the submission of the first version of the present paper), which essentially reported on an application of metamorphic exploration introduced in Section 7.2.2.6.

metamorphic exploration has the following limitations: (1) It is a dynamic approach, and hence cannot be applied if the executable program is not yet available. (2) As an MR involves multiple executions of the program, our approach must run the program at least twice, which could be an issue in situations where the execution time is very long. (3) It depends on the quality of the chosen MRs, and an MR may not necessarily cover all possible inputs or features. For example, to apply the MR defined in Section 6.1, we need to avoid one-way traffic, which is an obvious limitation of this MR's applicability. To enhance test coverage, other MRs need to be developed to allow one-way traffic, such as those reported by Brown et al. [34]. (4) It depends on the quality of the chosen input—this is a limitation of all test-case-based techniques including software testing and other dynamic approaches. (5) Its effectiveness depends on the background, experience, and knowledge of the users who conduct the metamorphic exploration. (6) As a user-oriented technique, metamorphic exploration is *not* a systematic approach in the sense that it is not designed to exhaustively cover all aspects of a software system—it only covers those aspects that the particular user is interested in: Different users may draw different conclusions based on their own results.

Regarding the symmetry MRP: Although we believe that symmetry is a pervasively desirable property for many computer systems, our empirical studies reported in this paper only involved the following five narrow areas: commercial websites, navigation services, location-based search, image analysis, and video analysis. Therefore, there is a threat to the external validity of our results, that is, a threat to the generalization of our findings to other areas that have not yet been investigated.

Are there areas where symmetry cannot be applied? We do not know the answer but cannot exclude the possibility. However, even in such situations, attempting to conduct metamorphic exploration with the idea of symmetry would still be useful because such an attempt would allow the user to realize that the system is asymmetric. In any case, the external validity of our results can only be enhanced through additional empirical studies involving different software features and different application domains.

An important issue that has not been addressed in this paper relates to the selection of the most suitable symmetry MRs when there are multiple candidates. For some systems, multiple symmetry MRs may be identified, but they may not all be equally useful. Consider again the problem of finding a shortest path in an undirected graph, discussed in Section 2, for which a number of symmetry MRs could be identified: Apart from the symmetry MR "swapping the origin and destination nodes should not affect the length of the shortest path," we can have others, such as [25]: Let $(G,\ a,\ b)$ be the *source input*, and $(G',\ a',\ b')$ be the follow-up input, where G' is a permutation (isomorphic graph) of G, the node a' in G' corresponds to the node a in G, and the node b' in G' corresponds to the node b in G. Then the length of the shortest path between a and b in G should be the same as that between a' and b' in G'. Given these different types of MRs, all of which belong to the symmetry MRP, how could a user find the most suitable one? The users need practical guidance for MR selection and prioritization. This research question was studied in our previous work [25], [90] in the context of metamorphic testing, but not investigated in the present paper for the purpose of metamorphic exploration. Although we have given a general principle in Section 10.2—that selection of MRs for metamorphic exploration depends on the interests of individual users and stakeholders—more research is needed to develop more specific guidelines. Before this question can be properly addressed, there is a possibility that the user will select a less effective MR to perform metamorphic exploration and, consequently, obtain less useful results.

Although we did not address the issue of selection of the most suitable symmetry MRs, our previous work has indicated that, identifying the concept of diversity as a guide [30], it is possible to train testers with no previous MT experience to a level where they can quickly identify and apply effective MRs within a matter of hours [20]–[22]. We have seen testers newly trained in this way, even working as individuals, independently and in an ad hoc manner, identify MRs with very high fault-detection effectiveness, even finding previously undetected faults in extensively tested software [20]. Given this relative ease and speed with which ab initio industrial MT users can be prepared, we are confident of the probable equal facility to become conversant and effective with metamorphic exploration. Obviously, however, verification of this will require further studies of actual user training and evaluation, which is something we look forward to doing.

Finally, we wish to point out that the "change direction" MRIP has obvious limitations: First, it is possible that a user may not be able to find any "direction element" in the program's input. In this situation, "change direction" cannot be applied. Second, even if some direction elements can be identified, they may not necessarily cover the features that the user is actually interested in. For example, if the user is not interested in search and sort functions, then the MR presented in Section 5 related to price sorting would be useless. In this situation, other types of MRIPs might be desirable.

11.4   Related Industrial Research and Development

Although this research targets end users, our methodology (which makes use of MRs to enhance software understanding, use, and quality assurance) is useful for developers, too. Adobe Systems, for example, has confirmed the usefulness of MRs, with their feedback summarized as follows:

- The introduction of MRs into the software quality assurance process has resulted in a significant drop in the user complaint rate.
- Compared with conventional methods, the use of MRs has helped Adobe engineers to better understand the software under test and to more cost-effectively identify compatibility issues between systems and their environments.
- Adobe engineers have started to use MRs to facilitate profiling system performance for the Adobe Cloud Platform. They have found that, compared with conventional approaches, MRs have helped them to more accurately estimate the *operational readiness* of applications, including the budget needed to set up an application in the operational environment.

Recently, the usefulness of MRs in the context of verifying machine learning (ML) applications has been recognized by researchers from Accenture. They noted that "unlike testing of traditional applications, finding one (or a few) instances of incorrect classification from an ML application does not indicate the presence of a bug" and that conventional testing techniques, when applied to ML applications, are "very expensive in terms of time and cost" [91, p. 118]. They presented a solution based on MT. Furthermore, Accenture Labs has reported on their patent titled "Verifying Machine Learning through Metamorphic Testing" [92, p. 12], in which they state that their methodology "needs only a few test cases (or even just one) to identify bugs in ML applications, thereby reducing the cost of testing significantly."

Also recently, organizations such as the US National Institute of Standards and Technology, have found that MRs are useful for cybersecurity enhancement [33], [93]. In Aug 2018, Google acquired GraphicsFuzz, a spinout company from Imperial College London, to develop secure and reliable graphics drivers by making use of MT within the Android ecosystem [5], [32], [94], [95].

We plan to continue to collaborate with industry to investigate the implications of this research on a larger scale.

12 CONCLUSION

In traditional bespoke system development, developers are often assumed to know the user requirements well when they release the final software product, and the users also usually receive appropriate system specifications and training. This is quite different from the situation where software is developed for the mass market. Furthermore, in modern IT paradigms, the software product may not necessarily satisfy the needs of all users (who may have diverse objectives), and the users may not necessarily understand how the system really works, or whether or not the service is really suitable for their specific information needs.

We posed three research questions in Section 1, all of which we have addressed using the concepts of metamorphic relations, metamorphic relation patterns, metamorphic relation input patterns, and metamorphic exploration. We have conducted a series of empirical studies using a wide variety of web applications and open source and commercial off-the-shelf products, including 65 of the most popular commercial websites, Google Maps navigation, Google Maps location-based search, MATLAB, OpenCV, Facebook, and the Google Cloud Video Intelligence. The empirical results show that our metamorphic exploration solution is simple, applicable, and highly effective for all systems investigated, hence providing an affirmative answer to all three research questions.

We have also provided a critical analysis of the limitations of our approach. In particular, the proposed symmetry MRP may not necessarily be applicable to all programs or all aspects of a program, and there can be situations where multiple symmetry MRs exist and where users may find it difficult to prioritize them. In this paper, we have only provided a guideline, or direction, for metamorphic exploration, using the symmetry MRP as a case study. Some of the discussions presented in this paper may even involve an oversimplification of the problem being solved. Nevertheless, we have opened the door to a new research direction, and paved the way for more in-depth research in the future: We have provided the fundamentals (with sufficient detail), including the methodology, the main concepts, and a reasonably scaled empirical evaluation.

This research has focused on the symmetry MRP and the "change direction" MRIP. In future research, we will continue to look at the applications of these patterns, and study other types of patterns for the purpose of metamorphic testing and exploration.

ACKNOWLEDGMENTS

REFERENCES

[1] M. Pezzè and C. Zhang, "Automated test oracles: a survey," in *Advances in Computers*, A. Memon, Ed. Elsevier Science & Technology, 2014, vol. 95, ch. 1, pp. 1–48.

[2] R. A. P. Oliveira, U. Kanewala, and P. A. Nardi, "Automated test oracles: State of the art, taxonomies, and trends," in *Advances in Computers*, A. Memon, Ed. Elsevier Science & Technology, 2014, vol. 95, ch. 3, pp. 113–199.

[3] 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'14)*, 2014, pp. 216–226.

[4] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[5] A. F. Donaldson and A. Lascu, "Metamorphic testing for (graphics) compilers," in *Proceedings of the IEEE/ACM 1st International Workshop on Metamorphic Testing (MET '16), in conjunction with the 38th International Conference on Software Engineering (ICSE '16)*. ACM, 2016, pp. 44–47.

[6] D. C. Jarman, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing for Adobe data analytics software," in *Proceedings of the IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET '17), in conjunction with the 39th International Conference on Software Engineering (ICSE '17)*, 2017, pp. 21–27.

[7] M. F. Kıraç, B. Aktemur, and H. Sözer, "VISOR: A fast image processing pipeline with scaling and translation invariance for test oracle automation of visual output systems," *Journal of Systems and Software, Special Issue on Test Oracles*, vol. 136, pp. 266–277, 2018.

[8] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the IEEE/ACM 40th International Conference on Software Engineering (ICSE '18)*. ACM, 2018, pp. 303–314.

[9] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, 2016.

[10] M. M. Lehman, "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060–1076, Sep 1980.

[11] G. Melnik, F. Maurer, and M. Chiasson, "Executable acceptance tests for communicating business requirements: Customer perspective," in *Proceedings of AGILE 2006 Conference (AGILE '06)*. IEEE, 2006.

[12] F. Pastore, L. Mariani, and G. Fraser, "CrowdOracles: Can the crowd solve the oracle problem?" in *Proceedings of the IEEE 6th International Conference on Software Testing, Verification and Validation*. IEEE, 2013, pp. 342–351.

[13] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *Journal of Systems and Software*, vol. 126, p. 57–84, 2017.

[14] B. Korel, "Black-box understanding of COTS components," in *Proceedings the 7th International Workshop on Program Comprehension*. IEEE, 1999.

[15] A. Andrews, S. Ghosh, and E. M. Choi, "A model for understanding software components," in *Proceedings of the International Conference on Software Maintenance (ICSM '02)*. IEEE, 2002.

[16] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 684–702, 2009.

[17] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring program comprehension: A large-scale field study with professionals," *IEEE Transactions on Software Engineering*, DOI: 10.1109/TSE.2017.2734091.

[18] A. Gotlieb, "Exploiting symmetries to test programs," in *Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE'03)*, 2003, pp. 365–374.

[19] C. N. Ip and D. L. Dill, "Better verification through symmetry," *Formal Methods in System Design*, vol. 9, no. 1–2, p. 41–75, 1996.

[20] 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.

[21] D. Towey and T. Y. Chen, "Teaching software testing skills: Metamorphic testing as vehicle for creativity and effectiveness in software testing," in *Proceedings of the IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE '15)*. IEEE, 2015, pp. 161–162.

[22] D. Towey, H. Liu, T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "Metamorphic testing: A new student engagement approach for a new software testing paradigm," in *Proceedings of the IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE '16)*. IEEE, 2016, pp. 218–225.

[23] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.

[24] 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.

[25] T. Y. Chen, D. H. Huang, T. H. Tse, and Z. Q. Zhou, "Case studies on the selection of useful relations in metamorphic testing," in *Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC'04)*. Polytechnic University of Madrid, 2004, pp. 569–583.

[26] 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, 2018.

[27] M. Lindvall, D. Ganesan, R. Árdal, and R. E. Wiegand, "Metamorphic model-based testing applied on NASA DAT —an experience report," in *Proceedings of the IEEE/ACM 37th International Conference on Software Engineering (ICSE '15)*, 2015, pp. 129–138.

[28] 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.

[29] U. Kanewala, L. L. Pullum, S. Segura, D. Towey, and Z. Q. Zhou, "Message from the workshop chairs," in *Proceedings of the IEEE/ACM 1st International Workshop on Metamorphic Testing (MET '16), in conjunction with the 38th International Conference on Software Engineering (ICSE '16)*. ACM, 2016.

[30] T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Q. Zhou, "A revisit of three studies related to random testing," *Science China Information Sciences*, vol. 58, no. 052 104:1–052 104:9, 2015.

[31] J. Regehr, "Finding compiler bugs by removing dead code," http://blog.regehr.org/archives/1161, June 20, 2014.

[32] 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.

[33] 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.

[34] J. Brown, Z. Q. Zhou, and Y.-W. Chow, "Metamorphic testing of navigation software: A pilot study with Google Maps," in *Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS-51)*, 2018, pp. 5687–5696, available: http://hdl.handle.net/10125/50602.

[35] X. Xie, J. W. K. 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, pp. 544–558, 2011.

[36] S. Sailer. (Sep 20, 2016) Great moments in Google: "American inventors". [Online]. Available: http://www.unz.com/isteve/great-moments-in-google-american-inventors/

[37] Z. Q. Zhou, T. H. Tse, F.-C. Kuo, and T. Y. Chen, "Automated functional testing of web search engines in the absence of an oracle," Department of Computer Science, The University of Hong Kong, Tech. Rep. TR-2007-06, 2007.

[38] 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.

[39] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of RESTful web APIs," *IEEE Transactions on Software Engineering*, in press.

[40] F.-H. Su, J. Bell, C. Murphy, and G. Kaiser, "Dynamic inference of likely metamorphic properties to support differential testing," in *Proceedings of the IEEE/ACM 10th International Workshop on Automation of Software Test*. IEEE, 2015, pp. 55–59.

[41] 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, pp. 245–269, 2016.

[42] J. Troya, S. Segura, and A. Ruiz-Cortés, "Automated inference of likely metamorphic relations for model transformations," *Journal of Systems and Software, Special Issue on Test Oracles*, vol. 136, pp. 188–208, 2018.

[43] M. Lindvall, A. Porter, G. Magnusson, and C. Schulze, "Metamorphic model-based testing of autonomous systems," in *Proceedings of the IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET '17), in conjunction with the 39th International Conference on Software Engineering (ICSE '17)*, 2017, pp. 35–41.

[44] P. W. Anderson, "More is different," *Science*, vol. 177, no. 4047, p. 393–396, 1972.

[45] I. Stewart, *Symmetry: A Very Short Introduction*. Oxford University Press, 2013.

[46] A. Zee, *Fearful Symmetry: The Search for Beauty in Modern Physics*. Princeton University Press, 2016.

[47] M. Hargittai and I. Hargittai, *Symmetry through the Eyes of a Chemist*, 3rd ed. Springer, 2009.

[48] G. B. Arfken, H. J. Weber, and F. E. Harris, *Mathematical Methods for Physicists: A Comprehensive Guide*, 7th ed. Amsterdam: Academic Press, 2012.

[49] I. M. Yaglom, *Felix Klein and Sophus Lie: Evolution of the Idea of Symmetry in the Nineteenth Century*, 1st ed. Birkhäuser, 1990.

[50] E. B. Vinberg, *A Course in Algebra*. American Mathematical Society, 2003.

[51] D. J. Gross, "The role of symmetry in fundamental physics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 25, pp. 14 256–14 259, 1996.

[52] S. Das and G. Kunstatter, "The central role of symmetry in physics," *Journal of Applied and Fundamental Sciences*, vol. 2, no. 2, pp. 69–77, 2016.

[53] M. Bañados and I. Reyes, "A short review on Noether's theorems, gauge symmetries and boundary terms," *International Journal of Modern Physics D*, vol. 25, no. 10, pp. 1 630 021:1–1 630 021:74, 2016.

[54] D. I. Perrett, D. M. Burt, I. S. Penton-Voak, K. J. Lee, D. A. Rowland, and R. Edwards, "Symmetry and human facial attractiveness," *Evolution and Human Behavior*, vol. 20, pp. 295–307, 1999.

[55] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*, 2nd ed. Pearson, 2002.

[56] A. T. Collins, J. M. Rose, and S. Hess, "Interactive stated choice surveys: a study of air travel behaviour," *Transportation*, vol. 39, no. 1, pp. 55–79, 2012.

[57] A. M. Degeratu, A. Rangaswamy, and J. Wu, "Consumer choice behavior in online and traditional supermarkets: The effects of brand name, price, and other search attributes," *International Journal of Research in Marketing*, vol. 17, no. 1, pp. 55–78, 2000.

[58] K. Suk, J. Lee, and D. R. Lichtenstein, "The influence of price presentation order on consumer choice," *Journal of Marketing Research*, vol. 49, no. 5, pp. 708–717, 2012.

[59] Deloitte. (2017) Global powers of retailing 2017: The art and science of customers. [Online]. Available: https://www2.deloitte.com/global/en/pages/consumer-business/articles/global-powers-of-retailing.html

[60] Alexa. (2017) Top sites by category: Shopping/Health/Pharmacy/Online Pharmacies. [Online]. Available: http://www.alexa.com/topsites/category/Shopping/Health/Pharmacy/Online_Pharmacies

[61] eBizMBA: The eBusiness Guide. (2017) Top 15 most popular real estate websites. [Online]. Available: http://www.ebizmba.com/articles/real-estate-websites

[62] ——. (2017) Top 15 most popular car websites. [Online]. Available: http://www.ebizmba.com/articles/car-websites

[63] ISO/IEC 25010:2011, "Systems and software engineering – systems and software quality requirements and evaluation (SQuaRE) – system and software quality models," 2011.

[64] Quora, Inc. (2012) Why does Google only return 50 pages of 10 results when it claims that there are 560,000 results? [Online]. Available: https://www.quora.com/

[65] D. Feinleib, *Bricks to Clicks: Why Some Brands Will Thrive in E-Commerce and Others Won't*. Apress, 2017.

[66] A. Clarke, *SEO 2018: Learn Search Engine Optimization with Smart Internet Marketing Strategies*. CreateSpace Independent Publishing Platform, 2017.

[67] J. C. Wong, "Delivery robots: a revolutionary step or sidewalk-clogging nightmare?" *The Guardian*, Apr. 12, 2017. [Online]. Available: https://www.theguardian.com/technology/2017/apr/12/delivery-robots-doordash-yelp-sidewalk-problems

[68] A. Gesenhues, "Google searches now correspond to user location instead of domain," *Search Engine Land*, Oct. 27, 2017. [Online]. Available: https://searchengineland.com/google-searches-now-correspond-location-versus-country-services-attached-domain-285769

[69] Google, "Target ads to geographic locations," 2017. [Online]. Available: https://support.google.com/adwords/answer/1722043?hl=en

[70] P. Kabos and V. S. Stalmachov, *Magnetostatic Waves and Their Application*. Springer, 1994.

[71] F. Han, *A Modern Course in University Physics: Optics, Thermal Physics, Modern Physics*. World Scientific, 2017.

[72] H. Kragh, "Contemporary history of cosmology and the controversy over the multiverse," *Annals of Science*, vol. 66, no. 4, pp. 529–551, 2009.

[73] D. Deutsch, *The Beginning of Infinity: Explanations That Transform the World*. Viking, 2011.

[74] J. Gribbin, *In Search of Schrödinger's Cat: Quantum Physics and Reality*. Random House Publishing Group, 2011.

[75] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*, 1st ed. Cambridge University Press, 2008.

[76] ISO/IEC/IEEE 24765:2010(E), "Systems and software engineering – vocabulary," 2010.

[77] F.-F. Li. (2017) Announcing Google Cloud Video Intelligence API, and more cloud machine learning updates. Google Cloud Big Data and Machine Learning Blog. [Online]. Available: https://cloud.google.com/blog/big-data/2017/03/announcing-google-cloud-video-intelligence-api-and-more-cloud-machine-learning-updates

[78] Google, Inc. (2017) Cloud video intelligence - video content analysis. [Online]. Available: https://cloud.google.com/video-intelligence/

[79] P. E. Ammann and J. C. Knight, "Data diversity: An approach to software fault tolerance," *IEEE Transactions on Computers*, vol. 37, no. 4, pp. 418–425, 1988.

[80] A. Carzaniga, A. Gorla, N. Perino, and M. Pezzè, "Automatic workarounds: Exploiting the intrinsic redundancy of web applications," *ACM Transactions on Software Engineering and Methodology*, vol. 24, no. 3, pp. 16:1–16:42, 2015.

[81] X. Zhang, Y. Gao, and M. K. H. Leung, "Recognizing rotated faces from frontal and side views: An approach toward effective use of mugshot databases," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 4, pp. 684–697, 2008.

[82] V. I. Stoichita, *A Short History of the Shadow*. Reaktion Books, 1997.

[83] L. Shan and H. Zhu, "Generating structurally complex test cases by data mutation: A case study of testing an automated modelling tool," *The Computer Journal*, vol. 52, no. 5, pp. 571–588, 2009.

[84] 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*. IEEE, 2015, pp. 8–15.

[85] Lao Tzu, *Tao Te Ching*. Wordsworth Editions Limited, 1997, translated by Arthur Waley.

[86] S. Segura and Z. Q. Zhou, "Presentation slides for ICSE 2018 Technical Briefing on metamorphic testing," 2018. [Online]. Available: http://doi.org/10.5281/zenodo.1256230

[87] M. Pezzè and M. Young, *Software Testing and Analysis: Process, Principles, and Techniques*. New York: Wiley, 2008.

[88] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, 2004.

[89] P. Rodeghero, S. Jiang, A. Armaly, and C. McMillan, "Detecting user story information in developer-client conversations to generate extractive summaries," in *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering (ICSE '17)*, 2017.

[90] Y. Cao, Z. Q. Zhou, and T. Y. Chen, "On the correlation between the effectiveness of metamorphic relations and dissimilarities of test case executions," in *Proceedings of the 13th International Conference on Quality Software (QSIC '13)*. IEEE, 2013, pp. 153–162.

[91] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '18)*. ACM, 2018, pp. 118–128.

[92] Accenture. (2018) Quality engineering in the new: A vision and R&D update from Accenture Labs and Accenture Testing Services. [Online]. Available: https://www.accenture.com/t20180627T065422Z__w__/us-en/_acnmedia/PDF-81/Accenture-Quality-Engineering-POV.pdf

[93] N. Mouha, M. S. Raunak, D. R. Kuhn, and R. Kacker, "Finding bugs in cryptographic hash function implementations," *IEEE Transactions on Reliability*, in press.

[94] GraphicsFuzz homepage. [Online]. Available: https://www.graphicsfuzz.com

[95] GraphicsFuzz. How it works. [Online]. Available: https://www.graphicsfuzz.com/howitworks.html
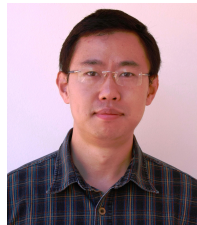
Zhi Quan Zhou received the BSc degree in computer science from Peking University, China, and the PhD degree in software engineering from The University of Hong Kong. He is currently an associate professor in software engineering at the University of Wollongong, Australia. His current research interests include software testing and debugging, software quality assessment, user experience improvement, and citation analysis. Zhou was a main contributor to some of the earliest research papers on metamorphic testing, and was one of the few pioneers who opened up and established this research field. He co-founded the ICSE International Workshop on Metamorphic Testing in 2016 (MET '16), and was an invited keynote speaker at ICSE MET '17. Zhou was selected for a Virtual Earth Award by Microsoft Research, Redmond.

Liqun Sun received the BSc and MSc degrees in physics from Donghua University, China. He is currently an MPhil student in computer science at the University of Wollongong, Australia. He worked as a software engineer at Tencent Technology and Taiwan Semiconductor Manufacturing Company. His current research interests include software testing and analysis.

Tsong Yueh Chen received the BSc and MPhil degrees from The University of Hong Kong; MSc degree and DIC from the Imperial College of London University; and PhD degree from The University of Melbourne. He is a professor of software engineering at Swinburne University of Technology, Australia. Prior to joining Swinburne, he had taught at The University of Hong Kong and The University of Melbourne. His current research interests include software testing and analysis. He is the inventor of adaptive random testing and metamorphic testing.

Dave Towey received the BA and MA degrees from The University of Dublin, Trinity College; PgCertTESOL from The Open University of Hong Kong; the MEd degree from The University of Bristol; and the PhD degree from The University of Hong Kong. He is an associate professor in the School of Computer Science, University of Nottingham Ningbo China (UNNC), where he serves as the school director of teaching and learning, and the deputy head. He also serves as deputy director of the International Doctoral Innovation Centre. His current research interests include software testing (especially adaptive random testing, for which he was amongst the earliest researchers who established the field, and metamorphic testing), computer security, and technology-enhanced education. He co-founded the ICSE International Workshop on Metamorphic Testing in 2016, and is a member of both the IEEE and the ACM.