* Metamorphic Testing*

1st Peng Wu
a State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
wp@ios.ac.cn

2nd Hepeng Dai
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China
daihepeng@sina.cn

3st Chang-ai Sun, *Senior Member, IEEE,*
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China
casun@ustb.edu.cn

***Abstract*—Metamorphic testing (MT) is a promising technique to alleviate the oracle problem, which fist defines metamorphic relations (MRs) used to generate new test cases (i.e. follow-up test cases) from the original test cases (i.e. source test cases). Both source and follow-up test cases are executed and their results are verified against the relevant MRs.**

***Index Terms*—metamorphic testing, control test process, partition**

I. Introduction

Test result verification is an important part of software testing. A test oracle [1] is mechanism that can exactly decide whether or not the output produced by a programs is correct. However, there are situations, where it is difficult to decide whether the result of the software under test (SUT) agrees with the expected result. This situation is known as oracle problem [2], [3].

Metamorphic testing (MT) [4], [5] is one of several techniques to alleviate oracle problem. MT first define metamorphic relations (MRs) through using some properties of SUT. Then, MRs are used to generate new test cases called follow-up test cases from original test cases known as source test cases. Next, both source and follow-up test cases are executed and their result are verified against the corresponding MRs.

II. Background

In this section, we present the basics to understand our approach. we start with a brief introduction to metamorphic testing, and then describe dynamic random testing.

A. Metamorphic Testing

A test oracle is a mechanism used to verify the correctness of outputs of a program [1]. However, there are test oracle problem [2], [3] in testing, that is, there are not an oracle or the application of such an oracle is very expensive. In order to alleviate the test oracle, several techniques have been proposed such as N-version testing [6], metamorphic testing (MT) [4], assertions [7], machine learning [8], etc. Among of them, MT obtains metamorphic relations (MRs) according to the properties of software under test (SUT). MRs are used to generate follow-up test cases from source test cases, then the both source and follow-up test cases are executed, and their results are verified against the corresponding MRs. If an MR is violated, that is, a fault is detected.

Let us use a simple example to illustrate how MT works. For instance, consider the mathematic function $f(x, y)$ that can calculate the maximal value of two integers x and y. There is a simple and obvious property: the order of two parameters x and y does not affect the output, which can be described as the follow metamorphic relation (MR): $f(x, y) = f(y, x)$. In this MR, (x, y) is source test case, and (y, x) is considered as follow-up test case. Letting P denotes a program that implements the function $f(x, y)$. Suppose P is executed with a test case $(1, 2)$, giving an output of 2. With respect to the MR $f(x, y) = f(y, x)$, P should next be executed with the other test case $(2, 1)$. Then the output of second execution is compared with that of the first test case: does $P(1, 2) = P(2, 1)$? If the equality does not hold, then we consider that P at least has one fault.

III. Empirical Study

We have conducted a series of empirical studies to evaluate the performance of M-AMT. The design of experiments is described in this section.

A. Research Questions

RQ1 How efficient and effective MT is at detecting faults of concurrent programs? Fault-detection efficiency and effectiveness are key criterions for evaluating the performance of a testing technique. In our study, we chose five real-life programs, and applied mutation analysis to evaluate them.

The National Natural Science Foundation of China (under Grant No. 61872039), the Beijing Natural Science Foundation (Grant No. 4162040), the Aeronautical Science Foundation of China (Grant No. 2016ZD74004), and the Fundamental Research Funds for the Central Universities (Grant No. FRF-GF-17-B29).

RQ2 How is the fault-detecting capability of each metamorphic relationship?

RQ3 How many metamorphic relations can detect most of faults existing in concurrent programs?

RQ4 How does the number of threads affect fault-detecting capability of MT?

RQ5 What is the actual test cases execution overhead when using MT to detect the faults of concurrent programs?

IV. Related Work

In this section, we describe related work of MT.

A. Metamorphic Testing

When testing a software system, the oracle problem appears in some situations where either an oracle does not exist for the tester to verify the correctness of the computed results; or an oracle does exist but cannot be used. The oracle problem often occurs in software testing, which renders many testing techniques inapplicable [2]. To alleviate the oracle problem, Chen et al. [4] proposed a technique named metamorphic testing (MT) that has been receiving increasing attention in the software testing community [2], [5], [9]. The main contributions to MT in the literature focused on the following aspects: i) MT theory; ii) combination with other techniques; iii) application of MT.

1 *Theoretical development of MT:* The MRs and the source test cases are the most important components of MT. However, defining MRs can be difficult. Chen et al. [10] proposed a specification-based method and developed a tool called MR-GENerator for identifying MRs based on category-choice framework [11]. Zhang et al. [12] proposed a search-based approach to automatic inference of polynomial MRs for a software under test, where a set of parameters is used to represent polynomial MRs, and the problem of inferring MRs is turn into a problem of searching for suitable values of the parameters. Then, particle swarm optimization is used to solve the search problem. Sun et al. [13] proposes a data-mutation directed metamorphic relation acquisition methodology, in which data mutation is employed to construct input relations and the generic mapping rule associated with each mutation operator to construct output relations. Liu et al. [14] proposed to systematically construct MRs based on some already identified MRs.
Without doubt, "good" MRs can improve the fault detection efficiency of MT. Chen et al. [15] reported that good MRs are those that can make the execution of the source-test case as different as possible to its follow-up test case. This perspective has been confirmed by the later studies [16], [17]. Asrafi et al. [18] conduct a case study to analyze the relationship between the execution behavior and the fault-detection effectiveness of metamorphic relations by code coverage criteria, and the results showed a strong correlation between the code coverage achieved by a metamorphic relation and its fault-detection effectiveness.
Source test cases also have a important impact on the fault detection effectiveness of MT. Chen et al. [19] compared the effects of source test cases generated by special value testing and random testing on the effectiveness of MT, and found that MT can be used as a complementary test method to special value testing. Batra and Sengupta [17] integrated genetic algorithms into MT to select source test cases maximising the the paths traversed in the software under test. Dong et al. [16] proposed a Path–Combination–Based MT method that first generates symbolic input for each executable paths and minis relationships among these symbolic inputs and their outputs, then constructs MRs on the basis of these relationships, and generates actual test cases corresponding to the symbolic inputs.
Different from the above investigates, we focused on performing test cases and MRs with fault revealing capabilities as quickly as possible by making use of feedback information. We first divided the input domain into disjoint partitions, and randomly selected an MR to generate follow-up test cases depended on source test case of related input partitions, then updated the test profile of input partitions according to the results of test execution. Next, a partition was selected according to updated test profile, and an MR was randomly selected from the set of MRs whose source test cases belong to selected partition.

2 *Combination with other techniques:* In order to improve the applicability and effectiveness of MT, it has been integrated into other techniques. Xie et al. [20] combined the MT with the spectrum-based fault localization (SBFL), extend the application of SBFL to the common situations where test oracles do not exist. Dong et al. [21] proposed a method for improving the efficiency of evolutionary testing (ET) by considering MR when fitness function is constructed. Liu et al. [22] introduced MT into fault tolerance and proposed a theoretical framework of a new technique called Metamorphic Fault Tolerance (MFT), which can handle system failure without the need of oracles during failure detection. In MFT, the trustworthiness of a test case depends on the number of violations or satisfactions of metamorphic relations. The more relations are satisfied and the less relations are violated, the more trustable test case is.

3 *Application of MT:* Sun et al. [23], [24] proposed a metamorphic testing framework for web services taking into account the unique features of SOA, in which MRs are derived from the description or Web Service Description Language (WSDL) [23] of the Web service, and on the basis on of MRs, follow-up test cses are generated depended on source test cases that are randomly generated according to the WSDL. Segura et al. [25] present a metamorphic testing approach for the detection of faults in RESTful Web APIs where they proposed six abstract relations called Metamorphic Relation Output Patterns (MROPs) that can then be instantiated into one or more concrete metamorphic relations. To evaluate this approach, they used both automatically seeded and real faults in six subject Web APIs.

V. Conclusion

In our future work, we plan to conduct experiments on more real-life programs to further validate the effectiveness of MT, and identify the limitations of our approach.

Acknowledgment

References

[1] E. J. Weyuker, "On testing non-testable programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.

[2] 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.

[3] K. Patel and R. M. Hierons, "A mapping study on testing non-testable systems," *Software Quality Journal*, vol. 26, no. 4, pp. 1373–1413, 2018.

[4] 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, Tech. Rep., 1998.

[5] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, p. 4, 2018.

[6] S. S. Brilliant, J. C. Knight, and P. Ammann, "On the performance of software testing using multiple versions," in *Proceedings of the 20th International Symposium on Fault-Tolerant Computing (FTCS'90)*, 1990, pp. 408–415.

[7] K. Y. Sim, C. S. Low, and F.-C. Kuo, "Eliminating human visual judgment from testing of financial charting software," *Journal of Software*, vol. 9, no. 2, pp. 298–312, 2014.

[8] W. K. Chan, S.-C. Cheung, J. C. Ho, and T. Tse, "Pat: A pattern classification approach to automatic reference oracles for the testing of mesh simplification programs," *Journal of Systems and Software*, vol. 82, no. 3, pp. 422–434, 2009.

[9] 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.

[10] T. Y. Chen, P.-L. Poon, and X. Xie, "Metric: Metamorphic relation identification based on the category-choice framework," *Journal of Systems and Software*, vol. 116, pp. 177–190, 2016.

[11] T. J. Ostrand and M. J. Balcer, "The category-partition method for specifying and generating fuctional tests," *Communications of the ACM*, vol. 31, no. 6, pp. 676–686, 1988.

[12] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei, "Search-based inference of polynomial metamorphic relations," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE'14)*, 2014, pp. 701–712.

[13] C.-A. Sun, Y. Liu, Z. Wang, and W. K. Chan, "μmt: A data mutation directed metamorphic relation acquisition methodology," in *Proceedings of the 1st International Workshop on Metamorphic Testing (MET'16), Co-located with the 38th International Conference on Software Engineering (ICSE'16)*, 2016, pp. 12–18.

[14] H. Liu, X. Liu, and T. Y. Chen, "A new method for constructing metamorphic relations," in *Proceedings of the 12th International Conference on Quality Software (QSIC'12)*, 2012, pp. 59–68.

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

[16] G. Dong, T. Guo, and P. Zhang, "Security assurance with program path analysis and metamorphic testing," in *Proceedings of the 4th IEEE International Conference on Software Engineering and Service Science (ICSESS'13)*, 2013, pp. 193–197.

[17] G. Batra and J. Sengupta, "An efficient metamorphic testing technique using genetic algorithm," in *International Conference on Information Intelligence, Systems, Technology and Management*, 2011, pp. 180–188.

[18] M. Asrafi, H. Liu, and F.-C. Kuo, "On testing effectiveness of metamorphic relations: A case study," in *Proceedings of the 15th International Conference on Secure Software Integration and Reliability Improvement (SSIRI'11)*, 2011, pp. 147–156.

[19] T. Y. Chen, F.-C. Kuo, Y. Liu, and A. Tang, "Metamorphic testing and testing with special values." in *Proceedings of the 5th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'04)*, 2004, pp. 128–134.

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

[21] G. Dong, S. Wu, G. Wang, T. Guo, and Y. Huang, "Security assurance with metamorphic testing and genetic algorithm," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'10)*, vol. 3, 2010, pp. 397–401.

[22] H. Liu, I. I. Yusuf, H. W. Schmidt, and T. Y. Chen, "Metamorphic fault tolerance: An automated and systematic methodology for fault tolerance in the absence of test oracle," in *Proceedings of the 36th International Conference on Software Engineering (ICSE'14)*, 2014, pp. 420–423.

[23] C.-A. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, and T. Y. Chen, "Metamorphic testing for web services: Framework and a case study," in *Proceedings of the 9th IEEE International Conference on Web Services (ICWS'11)*, 2011, pp. 283–290.

[24] ——, "A metamorphic relation-based approach to testing web services without oracles," *International Journal of Web Services Research*, vol. 9, no. 1, pp. 51–73, 2012.

[25] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of restful web apis," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1083–1099, 2018.