A Metamorphic Relation Identification Method based on GEP

Jie Hong[1], Jie Zhang[1], Qi Qiu[1], Angang Ma[1], Meng Li[1,2,3,*], Shiyu Yan[1,2,3], and Helin Gong[4]

[1]School of Computing, University of South China, Hengyang, Hunan, China
[2]Engineering and Technology Research Center of Software Evaluation and Testing for Intellectual Equipment of Hunan Province, Hengyang, Hunan, China
[3]CNNC Key Laboratory on High Trusted Computing, Hengyang, Hunan, China
[4]Nuclear Power Institute of China, Science and Technology on Reactor System Design Technology Laboratory, Chengdu, China
874201135@qq.com, 493017155@qq.com, quqihyy@163.com, 1095075717@qq.com
yanshiyu@usc.edu.cn, gonghelin06@qq.com,
*mlemon@usc.edu.cn

***Abstract* - Metamorphic testing is one of the effective methods to alleviate the test oracle problem. Metamorphic relation is the core of metamorphic testing, and there is no effective automatic identification technology. This paper transforms the metamorphic relation recognition issue into a symbolic expression regression problem. The test input pairs are generated using the preset input pattern, and the output pattern is mined by gene expression programming. After reduction and verification, a valid metamorphic relation is obtained. Experiments show that this method can identify the relation obtained by static analysis and a more significant number. At the same time, compared with existing technologies, it has apparent advantages in effectiveness, reliability, and performance.**

Keywords - test oracle; metamorphic relation identification; symbolic regression; gene expression programming.

I. Introduction

Software testing refers to selecting a set of test cases from the input domain of the software to be tested, using these test cases to execute the software, and comparing the actual output results with the expected output results. If not, it indicates a fault in the software to be tested.

We call the test oracle problem in test theory a phenomenon in which the execution result of a program cannot be predicted. For example, when testing the sin(x) function, we cannot accurately construct the expected results of sin(x), which makes the traditional test method with direct comparison ineffective.

Metamorphic Testing (MT) is one of the effective methods to alleviate the test oracle problems. MT uses Metamorphic Relation (MR) as the decision mechanism, and MR is the invariant relation between input and output. Metamorphic relation is the relation expected to follow between input and output when the target program is executed many times[1].

MR is the core of MT. The current MR identification technology can be divided into static analysis and dynamic mining according to whether the tested program is executed. The former usually obtain MR from mathematical and physical model properties, computational method characteristics, and program code specifications. The latter mining likely MR from a program running data because it does not require testers to have domain knowledge, so it has high engineering practice value.

There are three branches of the dynamic method at present:

1) Intelligent search method transforms MR identification into a numerical regression problem, such as MRI[2] and AutoMR[3]. They fix MR expression and solve coefficients.
2) Machine learning method transforms MR identification into a classification problem, such as ML[4]. It trains a machine learning model with a program flow chart and predicts the MR of the program under test.
3) From the perspective of symbolic expression regression, MR is mined on execution data and has no limitation on the format[5].

This paper establishes a general MR identification technology for scientific calculation programs. Major contributions include :

1) Propose an MR identification technology that can fully use prior knowledge, support rich symbolic expressions, and effectively identify the results.
2) The recognition process is stable. As long as MR exists, it must be found at most three times.
3) MR has high support on the test set and high recognition quality.

Note that, at present, it can only be used for scientific computing programs, and the identified MRs have been restricted to the format of input-only relation plus output-only relation.

The structure of this paper is as follows: In the second section, we introduce the background and motivation of MR recognition. In the third section, we present our method. In the fourth section, we design the experiment and submit the results in the fifth section. In the sixth section, we discuss the experimental effects. In the seventh section, we summarize and conclude this paper.

2767-6684/22/$31.00 ©2022 IEEE
DOI 10.1109/DSA56465.2022.00030

II. Background and Motivation

Dynamic MR identification technology has its limitations and shortcomings:

Intelligent search method: MR is limited to polynomial, MR expression is limited, and input pattern r and output pattern R are coupled to solve, which has a large amount of calculation. At the same time, the domain knowledge of the program under test is not considered. Many candidate solutions are meaningless to the test in the search space, further reducing efficiency.

Classification prediction method: based on the existing MR to predict, it is impossible to find new forms, it is difficult to find enough programs to form a training set in engineering, and the source code is also difficult to obtain.

Symbolic regression method: It is a data-driven approach—MR identification into a symbolic expression regression problem. Although MR form has no restrictions, the main challenge is constant numeric generation.

Gene Expression Programming (GEP) is a kind of swarm intelligence evolutionary algorithm proposed by C. Ferreira in 2001[6]. GEP is a popular and established evolutionary algorithm for automatically generating computer programs and mathematical models. It has found wide applications in symbolic regression, classification, automatic model design, combinatorial optimization, and real parameter optimization problems.

Studies have shown that GEP coding is simple, closed to genetic operations, and has fast evolution speed. In particular, the success rate of GEP is more significant than 0.66. As long as the expression exists, it can be found by executing GEP no more than three times[7]. GEP has achieved better results than traditional algorithms in symbolic regression, predicate association rule extraction, classification discriminant mining, time series prediction, and cellular automata evolution.

This paper assumes that MR can be expressed as a symbolic equation. Then the MR solution problem is transformed into an extended-expression regression problem. GEP technology is used to solve the symbolic regression.

III. Proposed Approach

MR is composed of input relation r and output relation R. In this paper, firstly, r is selected to generate test input, and then R is mined. The processing flow is shown in Figure 1. It divides into three sections. Specifically, MR is mined in Section III-A, the redundancy MRs are eliminated in Section III-B, and the final results are verified in Section III-C.

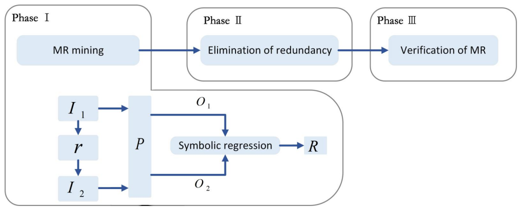

Fig. 1. The overall MR identification workflow.

The program under test is denoted as P, the input I, output O, O = P (I). Without losing generality, binary MR, namely, the invariant relation between the two executions, indicates the execution process of this method.

A. MR mining

Unlike the existing MR identification technology to solve r-R together, this method first presets the input relation r, then mines R.

1) Source input I_1: Randomly generated initial inputs on the input domain, each with n elements:

$$I_1 = i_1^{(1)} \quad i_1^{(2)} \quad i_1^{(3)} \quad \ldots \quad i_1^{(n)}$$

2) Input relation r: Taking elementary function as r;
3) Follow-up input I_2 : Select the r series to generate subsequent input I_2:

$$I_2 = r(I_1)$$

4) Get output: the output O_1, O_2 are obtained after the program under test have been executed with these inputs:

$$O_1 = P(I_1)$$
$$O_2 = P(I_2)$$

5) Mining R: Mining R on the output makes the fitting result $O_2^{'}$ of R (O_1) closest to O_2:

$$O_2^{'} = R(O_1); O_2^{'} \approx O_2$$

B. Elimination of redundancy

Because of the dynamical mining mean, there is redundancy MR, such as MRi : 2sin(x) = 2sin(2 * π + x), and this expression is simplified to sin(x) = sin(2 * π + x).

C. Verification of MR

Since the likely MR is obtained from the training data set, it still needs to be verified on the test data set. We used data-based probabilistic verification.

Construct a test set T, repeat the above process f times, and calculate the calculation degree of support s, if the score exceeds the support threshold h, the result is good, and MR is accepted.

1) Testing set T: Each pair of input and output constitutes a test record; m records form a set.
2) Calculation degree of support s: if MRi is established on the test record t_i, Count(true) plus 1, otherwise take the next record t_i+1 until T is empty, s = Count(true) / Count(T).
3) Number of verifications f: Perform verifying process f times by using different test sets T.
4) Support threshold h: If the Calculation degree of support s exceeds the threshold h, the verified result is successful. Otherwise, it fails.

D. Example

Based on the above algorithm, sin is an example to show the recognition process.

Assuming that we have implemented a program P, which evaluates the sin value (output) of the given input, identifies the relation r = (I_1, 2 * π + I_1) according to the prior knowledge.

And set the I_1 range of the primary input value to [0, 2 * π], and then generate the follow-up input I_2= (2 * π +I_1) through r. Taking I_1 and I_2as the input operation program P, we can get the output O_1 and O_2, where O_1= sin (I_1), O_2= sin (2 * π +I_1), drive the GEP mining and the relation R between them and eliminate redundancy. Then we can get R: sin (x) = sin (2 * π + x).

Finally, MRs are validated.

IV. Experimental Design

To explain the performance of GEP in automatic mining MR more clearly, this paper proposes three problems and designs two experiments for this purpose. In the first part of the experiment, GEP is applied to five representative functions for experiments. In the second part of the experiment, we compare GEP with AutoMR, a similar algorithm in recent research, in terms of solving efficiency, stability, and solving quality.

A. Research Questions

RQ1: Whether this method can obtain the MR of static analysis identification?

Since the artificial identification of MR has low stability, high risk, and no expansion, this paper proposes a dynamic method to identify MR. Our means are effective if MRs inferred by the static analysis can be identified.

RQ2: Can this method identify more MR than static analysis?

MR static analysis method highly depends on the domain knowledge of the software testing engineer. If this method can identify new MR which has not been reported by the static method, it has high productivity.

RQ3: Compared with the existing dynamic methods, which aspects of this method are better?

Compared with the existing dynamic methods from the aspects of efficiency, stability, quality, and diversity, the best applicable scenarios of various methods can be analyzed.

B. Experiment Settings

The Geppy is a computational framework implemented by GEP[8]. To better apply the experiment, It is necessary to set parameters reasonably.

1) GEP parameters
   a) Functional operator: Elementary function is the basic element of symbolic expression, and addition is used as the connector of expression. According to the prior knowledge of the problem domain, the operator can be defined, such as the Laplacian operator.
   b) Terminal node: Variable representation is generally used. Specific constants can be added based on prior problem domain knowledge, such as π for trigonometric functions.
   c) Population size and evolutionary algebra: Holding the default value for the GEP framework, both 100.
   d) The number of elitists: Only record the best three individuals found in all generations.
   e) Fitness function: MAE.
2) Testing set T: In the appropriate range, 100 random numbers are generated as the source input, and the follow-up input is calculated according to r to form a set of test input pairs. The number of elements in T is m = 100.
3) Test threshold h: The MR is accepted if the test result is less than 0.2. Otherwise, it is rejected.

C. Experiment design

Experiment 1: For RQ1, literature [2,9,10] derived a set of MR by mathematical properties. If this method can identify part or all of them, then RQ1 is true. The GEP method can obtain MR and is complementary to mathematical derivation methods.

For RQ2, if the MR identified is not mentioned in [2,9,10], then RQ2 is true, indicating that the GEP method has stronger searchability than the static analysis one.

Experiment 2: For RQ3, AutoMR is one of the latest MR dynamic identification techniques reported by literature, which achieved more efficiency than previous algorithms. To compare with AutoMR, We choose five representative functions in python's NumPy library to analyze efficiency, stability, quality, and diversity. These funcitions are arcsinh, arctan, ceil, log, and log10.

1) In terms of solving efficiency, we compare the time spent by a complete execution cycle of the two: both run 30 times and record and analyze the average execution time and the variance of the algorithm's execution time as the indicators of solving efficiency.
2) Relating to solving stability, since AutoMR uses PSO to search the MR relation, PSO as an approximate search algorithm can not find the optimal solution every time, so AutoMR may sometimes fail to see the results. This paper will compare the number of times AutoMR and GEP find the correct MR in 100 runs as the index of solving stability.
3) Concerning the solution quality, for the polynomial MR of different orders, AutoMR needs to be solved separately, and it cannot effectively remove the redundant MRs. For example, regarding MR1: sin (x) * sin (x) + sin (2 * π + x) * sin (2 * π + x) -2 * sin (x) * sin (2 * π + x) = 0, and MR2: sin (x) -sin (2 * π + x) = 0, it cannot remove redundant one.

4) The algebraic complexity[11] is employed as a metric for comparing the diversity of MRs. Since the input pattern is the same, the output pattern R is compared. The notation (mA, kM, gJ) denote the cost of m additions/subtractions, k multiplications/divisions and g algebra judgments (e.g. “<”, “>”, “≤”, “≥”, “=”). Algebra complexity of R could be denoted as: AC(R)=m+k+g. The large difference between the maximum and the minimum of algebraic complexity and the large variance in the MR set indicate that the MR set has a rich diversity.

V. Experimental Results

1) RQ1: Table I lists the functions we used in the existing literature, the corresponding MR for each function, and the input and output relation for each MR. We list the MRs equal to each other in this table to show that 15 MRs are successfully verified. This method can get the MR obtained by static analysis.

TABLE Ⅰ

GEP identifies MRs from paper

Numerical programs	MRs	Input Pattern	Output Pattern
sin[2][9]	MR1	$X^{'} = PI - X$	$Y^{'} = Y$
	MR2	$X^{'} = 2 * PI + X$	$Y^{'} = Y$
	MR3	$X^{'} = X$	$Y^{'} =- Y$
	MR4	$X^{'} = PI + X$	$Y^{'} =- Y$
	MR5	$X^{'} = 2 * PI - X$	$Y^{'} =- Y$
	MR11	$X^{'} = X - 2 * PI$	$Y^{'} = Y$
	MR12	$X^{'} = X - PI$	$Y^{'} =- Y$
	MR13	$X^{'} =- X - 4 * PI$	$Y^{'} =- Y$
	MR14	$X^{'} =- X - PI$	$Y^{'} =- Y$
	MR15	$X^{'} =- X - 3 * PI$	$Y^{'} =- Y$
	MR20	$X^{'} = 2 * X + 0.5 * PI$	$Y^{'} = (0.5 - Y * * Y) * 2$
Triangle square [10]	MR21	$a^{'} = b, b^{'} = a, c^{'} = c$	$Y^{'} = Y$
	MR22	$a^{'} = a, b^{'} = c, c^{'} = b$	$Y^{'} = Y$
	MR23	$a^{'} = c, b^{'} = b, c^{'} = a$	$Y^{'} = Y$
	MR24	$a^{'} = 2 * a, b^{'} = 2 * b, c^{'} = 2 * c$	$Y^{'} = 4 * Y$

2) RQ2: Table II shows the new MR we found using this method. We also use the test set to verify each MR mined, and the success rate of most MR detection is 100 %. The detection effect is generally good, and this method can obtain new and correct MR.

3) RQ3: Table Ⅲ compares the output pattern between GEP and AutoMR with the same input pattern, Figure 2-4 demonstrates the difference between the two methods' average execution time, variance, and support degree. To facilitate comparison, the logarithmic coordinates are adopted. It can be found that GEP has a shorter average time, minor variance, and higher support for MR recognition.

4) Figure 5 shows the difference between the maximum and minimum values of algebraic complexity and the variance of algebraic complexity of each MR set marked by GEP and AutoMR, respectively. The result illustrates that the difference between the maximum and minimum algebraic complexity of MR identified by GEP equals 1, and the variance is 0.4. The difference between the maximum and minimum MR algebraic complexity identified by AutoMR is 15, and the variance is 6.30555, both of which are larger than the former.AutoMR produces more diverse MR than GEP.

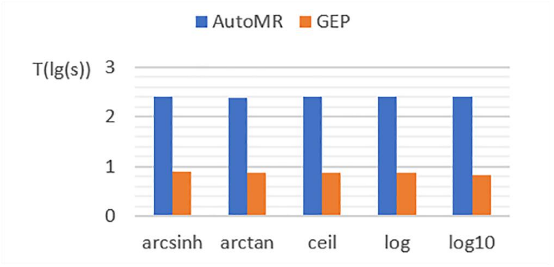

Fig. 2. Average execution time

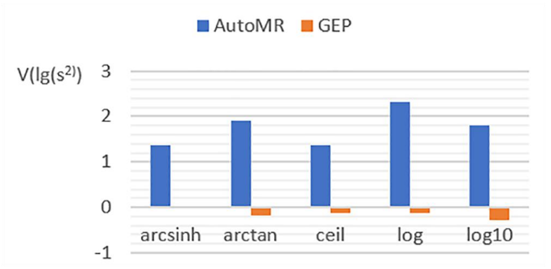

Fig. 3. The variance of running time

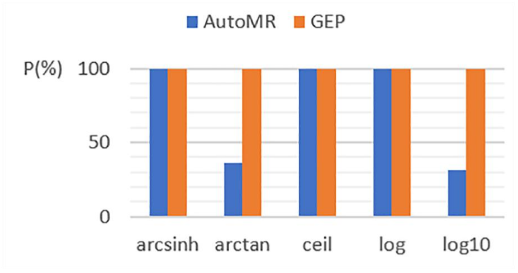

Fig. 4. Support degree score

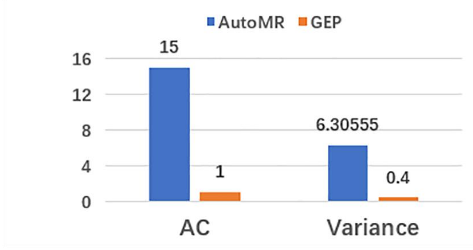

Fig. 5. Algebra Complexity of MR set identified by GEP and AutoMR

TABLE II
New MR identified by GEP

Numerical programs	**MRs**	**Input Pattern**	**Output Pattern**
sin[2][9]	MR6	$X_1' = (X_1 + X_2)/2$ $X_2' = (X_1 + X_3)/2$ $X_3' = (X_2 + X_3)/2$ $X_4' = X_1 + X_2 + X_3$	$Y_4' = 2 * Y_1' * Y_2' * Y_3' +$ $Y_3 * (5 * Y_1' * Y_1' - 3)$
	MR7	$X' = PI/2 - X$	$Y' = \sqrt{-2 * Y + 2 * (Y^Y)}$
	MR8	$X' = 3 * X$	$Y' = 2 * Y - 3 * Y * Y * Y$
	MR9	$X_1' = X_1 - X_2,$ $X_2' = X_1 + X_2$	$Y_2' = Y_1' * (3 * Y_1 * Y_1 - 8 * {Y_2}^4)$
	MR10	$X_1' = 3 * X,$ $X_2' = 5 * X$	$Y_2' = - Y_1' * Y_1'/Y + Y_1' + Y - 14$
	MR16	$X' = - 0.5 * X - 0.75 * PI$	$Y' = 0.5 - 0.5 * Y ** 0.5$
	MR17	$X' = X - 2.5 * PI$	$Y' = (Y ** 8 * (2.25 - 14.137166943 * Y)$ $+2)/(4 * 14.137166943 * Y - 2.25)$
	MR18	$X' = 2 * X - 1.5 * PI$	$Y' = 1 - 2 * Y * Y$
	MR19	$X' = - X - 2 * PI$	$Y' = - Y$
Triangle square [10]	MR25	$a' = \sqrt{2b^2 + 2c^2 - a^2},$ $b' = b,$ $c' = c$	$TriSquare(a', b', c') = 161 * TriSquare(a, b, c)/160$
	MR26	$a' = a,$ $b' = \sqrt{2a^2 + 2c^2 - b^2},$ $c' = c$	$TriSquare(a', b', c') = TriSquare(a, b, c) + 16 + 64/TriSquare(a, b, c)$
	MR27	$a' = a,$ $b' = b,$ $c' = \sqrt{2a^2 + 2b^2 - c^2}$	$TriSquare(a', b', c') = (TriSquare(a, b, c) ** 2 + 89$ $* TriSquare(a, b, c)/5 + 80)/(TriSquare(a, b, c)$ $+ 10)$
	MR6	$X_1' = (X_1 + X_2)/2$ $X_2' = (X_1 + X_3)/2$ $X_3' = (X_2 + X_3)/2$ $X_4' = X_1 + X_2 + X_3$	$Y_4' = 2 * Y_1' * Y_2' * Y_3' +$ $Y_3 * (5 * Y_1' * Y_1' - 3)$
	MR7	$X' = PI/2 - X$	$Y' = \sqrt{-2 * Y + 2 * (Y^Y)}$
	MR8	$X' = 3 * X$	$Y' = 2 * Y - 3 * Y * Y * Y$

TABLE III
The examples identified by GEP and AutoMR

Numerical programs	**MRs**	**Input Pattern**	**GEP Output Pattern**	**AutoMR Output Pattern**
arcsinh	MR6	$X' = - X$	$Y' = - Y$	$Y^2 + Y * Y' - 2 * Y - 2 * Y'^2 = 0$
arctan	MR7	$X' = - 2.732 - 2 * X$	$Y' = - Y$	$-3.113 + 2 * Y * Y' + 2 * Y + 2 * Y' = 0$
ceil	MR8	$X' = - 1.99 - X$	$Y' = - Y - 1$	$0.9999999999800762 - 2 * Y^2 + Y * Y * Y'$ $-Y * Y' + 2 * Y * Y' * Y' - Y + Y'^2 + Y'^3 + Y' = 0$
log	MR9	$X' = - 0.002 - X$	$Y' = Y$	$Y^2 + Y^3 + 2 * Y * Y * Y' - 2 * Y * Y'$ $+Y * Y' * Y' - Y + Y'^2 + Y' = 0$
log10	MR10	$X' = 2 * X$	$Y' = Y + 10/(10 * \pi + x)$	$-0.602 - 2 * Y + 2 * Y' = 0$

VI. Discussions

It can be seen from the experimental results that when compared with the static analysis method, it is found that this method can identify the existing MR and has a more vital solving ability.

Compared with the dynamic analysis methods such as AutoMR, we found that the solution efficiency, stability, and quality of this method were higher than those of the latter, the efficiency was high, and the quality of the identified MR test results was excellent. In the obtained MR expression, AutoMR fixes MR as a higher-order polynomial, and the result lacks sufficient reduction, making AutoMR superior to GEP in terms of algebraic complexity index. Therefore, it is necessary to study comprehensive diversity metrics to evaluate MR recognition more scientifically.

VII. Conclusion

This paper proposes an MR recognition technology based on GEP, transforming the transformation relation recognition problem into a symbolic expression regression problem. This method can identify the relation obtained by static analysis, has more quantity, has a more straightforward form, and can use prior knowledge.

Compared with the existing technologies such as AutoMR, it is found that the solution efficiency, solution stability, and solution quality are higher than those of the latter, and several new MR can be found in some tested programs. It is proven that the results are excellent and can be used as an extension.

In the future, we will consider extending the method to complex scenes, such as high dimension input and the piecewise functions. In addition, it is considered to improve the quality and performance of the recognition of this method.

Acknowledgment

This work is supported by the National Science and Technology Major Projects of China (No.2019ZX06005001),
National Natural Science Foundation of China (Grant No. 1 1905216, Grant No. 12175220), Natural Science Foundation of Hunan Province of China (No.2020JJ4519), Doctoral Research Startup Foundation of University of South China(No.220XQD050 and 220XQD049).

References

[1] T. Y. Chen, S. C. Cheung, and S. M. Yiu, “Metamorphic testing: a new approach for generating next test cases,” HongKong, 1998. doi: 10.48550/arXiv.2002.12543.

[2] J. Zhang et al., “Search-based inference of polynomial metamorphic relations,” in Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE 2014), 2014, pp. 701–712. doi: 10.1145/2642937.2642994.

[3] B. Zhang, H. Zhang, J. Chen, D. Hao, and P. Moscato, “Automatic Discovery and Cleansing of Numerical Metamorphic Relations,” in 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME 2019), Dec. 2019, pp. 235–245. doi: 10.1109/icsme.2019.00035.

[4] U. Kanewala and J. M. Bieman, “Using machine learning techniques to detect metamorphic relations for programs without test oracles,” in IEEE 24th International Symposium on Software Reliability Engineering (ISSRE 2013), 2013, pp. 1–10. doi: 10.1109/ISSRE.2013.6698899.

[5] L. Meng, W. Lijun, Y. Shiyu, and Y. Xiaohua, “Metamorphic Relation Generation for Physics Burnup Program Testing,” Int. J. Performability Eng., vol. 16, no. 2, pp. 297–306, Feb. 2020, doi: 10.23940/ijpe.20.02.p12.297306.

[6] C. Ferreira, “Gene Expression Programming: a New Adaptive Algorithm for Solving Problems,” Complex Syst., vol. 13, no. 2, pp. 87–129, Feb. 2001, doi: 10.1007/3-540-32849-1.

[7] Tang C J, ZHANG T, Zuo J, et al, “Knowledge discovery based on gene expression programming-history, achievements and future directions,” Computer Application, vol. 24, no. 10, pp. 7−10, 2004.

[8] Shuhua Gao, “geppy: a Python for gene expression programming,” Zenodo, Jun. 2020. https://github.com/ShuhuaGao/geppy (accessed May 09, 2022).

[9] T. Y. Chen, F.-C. Kuo, Y. Liu, and A. Tang, “Metamorphic Testing and Testing with Special Values,” in the 4th IEEE International Workshop on Source Code Analysis and Manipulation, 2004, pp. 128–134.

[10] D. Guowei, X. Baowen, C. Lin, N. Changhai, and W. LuLu, “Case st udies on testing with compositional metamorphic relations,” J. Southe ast Univ., vol. 24, no. 4, pp. 437–443, 2008.

[11] Z. W. Hui, S. Huang, H. Li, J. H. Liu, and L. P. Rao, “Measurable me trics for qualitative guidelines of metamorphic relation,” in Proceedin gs -
International Computer Software and Applications Conference, Sep. 2015, vol. 3, pp. 417–422. doi: 10.1109/COMPSAC.2015.179.