Software Testing LTAT.05.006

HW 7: Metamorphic Testing

Institute of Computer Science, University of Tartu

Instructions

- Submission deadline: Lab reports must be submitted within seven days. *For example, if your lab takes place on a Tuesday, you must submit your report no later than the following Monday, 23:59 hours.*
- Late submission policy: 50% of the total marks deducted for submissions up to 24 hours late; 100% of the total marks deducted for submissions more than 24 hours late
- Group: There should be a maximum of two members in a group. Answers should be your group work, explained in your own words. If you work in a group of two students, make sure to mention the names of both students in the lab report submitted.
- The maximum number of points is nine (9).

Introduction

Metamorphic Testing (MT) is a software testing technique to counter the test oracle problem—the challenge of differentiating a program's correct behavior from potentially incorrect behavior. Instead of checking each output individually, MT runs the program multiple times with changed inputs and compares the relationships between the outputs. These relationships, called **Metamorphic Relations** (MRs), ensure the program behaves as expected under different inputs. MT is useful for complex systems where expected results are hard to define because it doesn't require knowing the correct output—only that the relationships between outputs hold true.

MT has applications in various domains, where traditional testing methods fall short:

- **Scientific computing**: MT helps in validating the software's functionality in applications and calculators where the results are complex, or the exact values are unknown.
- **Machine learning and AI**: In machine learning models, predicting the exact output is hard and time-consuming. MT can be used to verify the consistency and correctness of these models.
- **APIs**: MT is especially useful when the web services are black-box, and the inner workings are unknown. For example, if a search API returns results for the query "best restaurants," adding a filter like "open now" should produce fewer results but still include

the original matches if they meet the filter criteria.

In this lab, you will be using MT to test black-box braking distance calculators and traffic sign recognition models to identify bugs in these systems.

Learning Objective

The exercise aims to give an understanding of metamorphic testing and its value. You will learn to use metamorphic testing principles in different scenarios and to identify metamorphic relationships that can help test complex software without needing to know the exact result. You will get an understanding of how metamorphic relations can be used to judge the performance of AI models.

Part A: Lab session

Task T1

In the first part of the homework, you will be testing different braking distance calculators. You will be given the first metamorphic relation along with the test as an example.

Setting up T1:

1. The lab materials contain a ZIP file Lab7.zip. Unzip the file and find the folder “task1”.
2. Either:
   a. Open the folder “task1” as an IntelliJ project;

   or:

   b. Go to IntelliJ. Click on File -> New -> Project from existing sources;
   c. Select the “task1” folder;
   d. Choose Maven for the Import Project prompt in IntelliJ and open the project.
3. The test suite is found under task1/src/test/java/CalculatorTestSuite.java. There you will be creating tests for the calculators.
4. Click Setup SDK to configure the project if necessary. Select JDK-21, however, it should work with any JDK version 19 or newer.
5. You can run the program in the IntelliJ terminal with the command: **java -jar lib/Calculators.jar**. Experiment with the different calculators and input values to see how they change.

Note: The program requires decimal numbers to be input using a dot (e.g., 0.05). If your computer's regional settings are configured for Estonia, the default decimal format uses a comma (",") instead of a dot ("."). This will result in an error message stating: "Please enter a valid number." To avoid this issue, ensure that your regional settings are adjusted to use a dot (".") as the decimal separator rather than a comma (",").

Look at the task description in Part B Task 1. Here is an example solution to find the first metamorphic relation (friction).

We know that braking on ice takes longer to come to a full stop than it would on dry asphalt. From this, we can reason that the braking distance should increase if the friction is decreased. To test if friction is handled correctly in the calculator, you should calculate the braking distance twice, once with lower friction and once with higher, and see if the distance changes accordingly. **Don’t change more than one parameter at a time,** because it will become difficult to determine which specific change caused the difference in the output, making it harder to verify the correctness of the MR being tested.

MR1: Driving on slippery roads is dangerous since the car takes longer to come to a stop. Switching from a high-friction surface (like dry asphalt) to a low-friction surface (ice) should increase the braking distance. Adding a constant $c < 0$ to μ (decreasing friction due to switching from "dry" to "icy") increases the braking distance s.

Mathematical function:

Input 1: $(t1, v1, G1, \mu1, B1)$
Output 1: $s1 = f(t1, v1, G1, \mu1, B1)$
Input 2: $(t2, v2, G2, \mu2, B2)$ with $t2=t1, v2=v1, G2=G1, \mu2=\mu1 + c, B1=B2$
Output2: $s2 = f(t1, v2, G2, \mu + c, B2)$
MR1: if $c > 0$ then $s1 > s2$
if $c < 0$ then $s1 < s2$

The corresponding test can be found in the file "CalculatorTestSuite.java". Run the test class and see which calculators fail this relation.

In the homework, continue identifying relations for all other parameters and improve the test suite to find all the bugs.

Note: The relationship described in this MR (braking distance being inversely related to friction) applies to both increasing and decreasing friction, as they are fundamentally the same relationship. Whether friction decreases (e.g., c<0) and braking distance increases, or friction increases (e.g., c>0) and braking distance decreases, the idea remains consistent. Thus, this relationship does not need to be redefined or repeated later in the homework for different values of c, as ***repeated relations will be counted as one and will not get points****.*

Task T2

In the second part of the homework, you will be testing 4 traffic sign classification models. During the lab, set up the models and testing class in a Python IDE of your choice. After you will see an example of how metamorphic testing can work for image classification.

MT in image classification differs from systems like the braking distance calculator by focusing on the model's performance rather than the relations of the output values. When using MT for image classification, transformations on images (based on MRs) are used to simulate real-world conditions. Then the models performance is evaluated based on how consistently it predicts the labels despite these transformations.

Setting up T2:

1. In the Lab folder, find the folder "**task2**".
2. In VSCode, select File -> Open Folder -> C:\.....\Metamorphic Testing\task2 (replace this with the path of your folder)
3. Make sure you have Python installed. For this task, you should use Python versions 3.11. Python version 3.13 will not work. You can check your Python version with "python --version".
4. Open the terminal in the IDE and create a virtual environment with the command **"py -3.11 -m venv env"** and activate the virtual environment with **".\env\Scripts\activate"**. This is optional but recommended to avoid any compatibility issues. Run the file **"python setup.py"**. This ensures that all required libraries are installed. You can later install any additional libraries that you wish to use to solve the task.

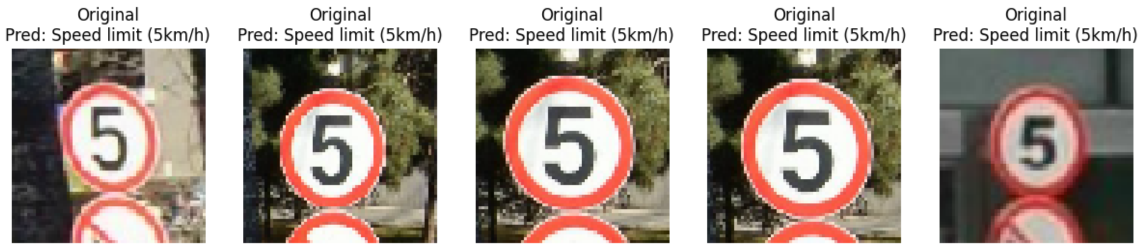
   (If you have any issues, the following versions are known to work without compatibility problems: Tensorflow version 2.18.0, Numpy 2.2.1, Pandas 2.2.3, Matplotlib 3.10.0, Scikit-learn 1.6.1, Scipy 1.15.1, Pillow 11.1.0)
5. Under the "**tests**" folder, you will find the "testing.py" file containing an example of how to modify the images by rotating the test data. The code will display the original and modified images and print out the consistency score table.
6. Run the "**testing.py**" file to ensure it works. Either run it from the Run button or in the IDE terminal with the command "python tests/testing.py". (Make sure you are in the correct directory, otherwise the command might be different.)

Example of the first relation:

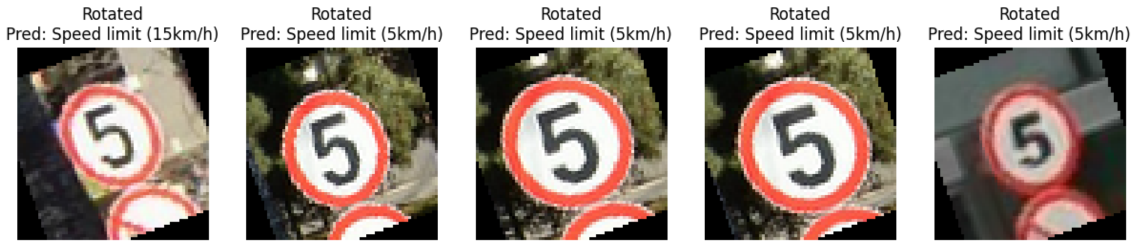
MR1 (General modification: Rotating the images

Explanation: Slightly rotating the images, for example by 20° should not significantly impact the predictions. Traffic signs or the camera can be tilted, which could cause problems with classifying the signs, so the model should work well even in this case.

Original images:


Original
Pred: Speed limit (5km/h)
Original
Pred: Speed limit (5km/h)
Original
Pred: Speed limit (5km/h)
Original
Pred: Speed limit (5km/h)
Original
Pred: Speed limit (5km/h)

Images after applying MR1:


Rotated
Pred: Speed limit (15km/h)
Rotated
Pred: Speed limit (5km/h)
Rotated
Pred: Speed limit (5km/h)
Rotated
Pred: Speed limit (5km/h)
Rotated
Pred: Speed limit (5km/h)

After looking at the first MR, see how it is implemented in the "testing.py" file. In the homework, you will have to think of 5 more metamorphic relations and make tests based on them.

Part B: Homework

Task T1: Simple Metamorphic Testing (4 points)

T1 Introduction

A self-driving car manufacturer M needs a system that calculates the braking distance of a car. Even a small error in the calculations could have disastrous results, so he needs to ensure it works correctly under all conditions. M is presented with four different calculators from suppliers and wants to select the best one.

The calculator considers **five parameters**: the type of road (friction), the slope of the road, the reaction time of the driver, the vehicle's speed, and the braking force. It is expected to work under a wide range of real-world scenarios, from highways with dry asphalt to icy neighborhood roads. The inner workings of the calculators are not known to M – it is impossible to verify the correctness of results with regular testing since the test oracle is unknown. With metamorphic testing, however, any inconsistencies will point to a problem with the calculators.

Your task is to **identify the metamorphic relations** (MRs) of the calculators based on the parameters the calculators use, and common sense. Experiment with different inputs with the calculators to get an idea of the outputs.

Parameters:

- t – Reaction time in seconds;
- v – Speed of the car in km/h. *Hint: speed is not a linear component in the function. Test out the calculators with very high speeds to see this effect.*
- G – Grade (slope) of the road, expressed as a decimal. Positive for an uphill grade and negative for a downhill road (values in the range of -1.0 to 1.0, where 0 is a flat road and 0.5 is a 50% incline);
- μ – Coefficient of friction between the tires and the road. It is assumed to be 0.7 on a dry road and between 0.3 and 0.4 on a wet road (values from 0 to 1.0).
- B – Braking strength, 0-1, where 0 is not pressing the brake and 1 is pressing the brake to the floor.

Signature for the formula: $f(t,v,G,\mu,B) \rightarrow s$ (stopping distance)

T1 Description

T1.1 You should think of **4 additional metamorphic relations, 1 relation per parameter**. **Explain** your thought process behind the relations, and why you think they hold. **Don't** define the same metamorphic relation multiple as described in the lab task. Write down the metamorphic relations as **mathematical functions**.

T1.2 Create a test suite based on the identified metamorphic relations so that all of the calculators are tested. You should create **at least 4 more tests, at least 1 for each relation**, not counting the example. You will get points for tests that **reveal a bug**.

T1.3 Find all of the bugs in the calculators. Write down the bugs found in each calculator in a table form. (See Appendix A)

T1.4 Explain which calculator the client should use (if any).

The first relation and test are given as an example in the Lab task.

Task T2: Metamorphic Testing on Machine Learning (5 points)

T2 Introduction

Now, M needs a traffic sign recognition system for his car. He has been offered different machine learning models by four sellers, each is said to be performing well. He wants to produce cars for three different markets, so he needs to select the models best suited to the specific conditions and needs of each area of the world. However, with limited funds, M must prioritize certain features and make some sacrifices, as he cannot afford to buy the ideal solution for all conditions. He should choose 3 out of the 4 models based on which model performs best in the given conditions.

Each market has specific requirements based on their region's driving conditions:

1. **Kenya**: Since many roads are unpaved and there are arid regions with dry, dusty conditions, the self-driving car needs a system that is highly resistant to blurry or fuzzy images caused by dirt on the signs and cameras as well as dust in the air.
2. **Sweden**: Since the winters in Sweden are long and it is often dark and snowy, the car here needs the traffic sign recognition system to function effectively in low-light conditions and during snowfall.
3. **California**: Since the weather in California is usually sunny and clear, the model needs to function especially well in brightly lit and vibrant conditions. Handling rain and snow is not as important as it is rare here.

Obviously, for real-life use, the model would need to perform reliably across all conditions, but for our hypothetical customer M, we will focus on prioritizing the specific requirements of each market.

The sellers have tested their models and reported them to have the following F1 scores:

System	**F1 score as reported by the seller**
Model A	0.860
Model B	0.845
Model C	0.851
Model D	0.849

As a reminder, the highest possible value of an F1 score is 1.0, meaning perfect precision and recall, and the lowest possible value is 0. The higher the F1 score, the better the model.

T2 Description

T2.1 Define at least 5 metamorphic relations:

Your job is to think of metamorphic relations (modifications on the test dataset) to test the models. The goal is to find out **which model is best for which market** by modifying the images to resemble the conditions of the different environments. **Explain your reasoning** for each transformation. An example of an MR is given in the Lab task.

You should think of at least **5 MRs** (not counting the example). If you make a modification that is market-specific, mention **which market** it is for. You should think of enough market-specific modifications so that your reasoning for assigning the models to the different markets is clear. Some can be general modifications that simulate real-world scenarios that would apply to all of the markets, like rotating an image slightly to simulate different angles of observation. These changes are to test the general robustness of the models, not to necessarily identify the best one for each market.

T2.2 Implement the MRs:

Create the modifications in the "testing.py" file. Each modification should be a separate function, there should be at least **5 different functions** that modify the images. Run the tests to see the changes in predictions for each modification. **Provide samples of the modified images in the report.**

Be reasonable in your modifications. If you change the image too much, it is expected that the model will be very inaccurate. If you can't tell what the sign is supposed to be, the model probably will not be able to either. You can ensure your changes are valid by having the test code show a few of each of the modified images.

Tips:

- *You can use plt.show to display the images after modifying, you don't have to save the modified images.*
- *The Python Image Library and NumPy might help with applying modifications.*

T2.3 Report the results:

To evaluate how robust each model is under these MRs, you will use the provided testing.py file. This calculates label consistency, which measures what percentage of predictions remain unchanged after applying each image transformation. Write down the results in a **table** form (1 table per model, see Appendix B), showing the changes in **label consistency** of each model during testing.

A decrease in label consistency could indicate that the model has improved by correcting previous misclassifications, but more often, it suggests a drop in robustness and accuracy. To determine whether the change is beneficial or detrimental, the tester could manually review the images with differing predictions. While this requires some effort, it is still significantly easier than reviewing the entire dataset of unlabeled images. For this homework task, however, you will not need to do this and can assume that generally the models' predictions get worse as the consistency drops.

Analyze each model based on the test results, explain each model's strengths and weaknesses, and say which model should be used in which market. **Explain your reasoning.**

Report

The report should be submitted in a ZIP file via the course web page selecting Lab 7. Your ZIP file should contain:

- The completed test suite for Task 1 (the file "CalculatorTestSuite.java")
- The completed test suite for Task 2 (the file "testing.py")
- A PDF file with the following content:
    - The name of the lab (Metamorphic Testing), the names of ALL group members, and the university IDs of ALL group members;

  Task 1:
    - **At least 4** metamorphic relations for task 1 with the mathematical functions;
    - A failure report table for Task 1 and the analysis of the results (see Appendix A);
    - The results of the testing (which calculator the client should choose and why);

  Task 2:
    - **At least 5** modifications along with the sample images and an explanation for the modifications;
    - A total of **4 tables** showing the changes in predictions throughout the testing (1 per model, see Appendix B)
    - The analysis of the testing results – explaining which model is best for which market and mentioning any other bugs or improvements that should be done for the models.

Grading

You can get up to 9 points for this lab. The grading is as follows:

Task 1 (4 points):

- T1.1: 2 points for correctly identifying the MRs and writing down the mathematical functions.
- T1.2: 1 point for the test suite. (0.25 per test that reveals a bug)
- T1.3: 0.5 point for the table.
- T1.4: 0.5 points for providing the right answer with a reasonable explanation.

0 points for T1.2, T1.3, and T1.4 if the test suite is not submitted.

Task 2 (5 points):

- T2.1: 3 points for modifying the datasets, providing the **sample images,** and explaining the choices of **how and why** you chose the specific modifications. (0.6 per modification - 0.3 for the code, 0.3 for the explanation.

- T2.2: 1 point for the filled tables. (0.25 per table)
- T2.3: 1 point for analyzing the results and assigning the models to the markets correctly.

0 points for the tasks if example images or testing file are not provided.

Appendix A

Sample Bug Report

Test Nr	**Tested Bug (MR used)**	**Calculator 1**	**Calculator 2**	**Calculator 3**	**Calculator 4**
TestMR1	MR 1 (friction)	Pass/Fail	Pass/Fail	Pass/Fail	Pass/Fail

Appendix B

Sample Model Report

Model X

MR #	**Metamorphic Relation**	**Market**	**F1 Score**	**Label Consistency (%)**
1	Image rotation	General		