Automated MATLAB Test Generation Based on Source Code

Introduction

Automated test generation is a crucial aspect of software development, as it helps ensure the correctness and reliability of the code. In the context of MATLAB, automated test generation plays a vital role in verifying the functionality and performance of MATLAB source code. This report explores the concept of automated MATLAB test generation based on source code and provides an overview of the available tools and techniques.

Test Case Generation in MATLAB

Test case generation in MATLAB involves the creation of input values and expected output values to validate the behavior of the code under test. MATLAB provides several tools and techniques to generate test cases based on source code. These tools aim to achieve different testing criteria, such as model coverage, decision coverage, and custom code analysis.

Simulink Design Verifier

Simulink Design Verifier is a powerful tool provided by MATLAB for test case generation. It extends the existing model coverage information from requirements-based tests and generates additional sequences of test inputs to meet the coverage objectives not satisfied during requirements-based testing. Simulink Design Verifier uses exhaustive searching techniques to generate input data that satisfies the analysis objectives. It can generate test cases for model decision coverage, custom code in a model, analyzable model components, and complete coverage of generated code.

EcoCoder-AI

EcoCoder-AI is an automatic code generation MATLAB/Simulink library that links directly to the target controller. It provides features like code generation, compilation, and one-click generation of executable files. EcoCoder-AI allows developers to convert Simulink model-based control strategies into ROS-based executable programs for the target controller. This tool simplifies the development of ROS-based applications using a model-based programming approach.

MATLAB Test Bench

MATLAB Test Bench is another tool provided by MATLAB for test case generation. It allows developers to write scripts or functions to test the algorithm in their MATLAB design function. The test bench varies the input data to simulate real-world conditions and checks if the output data meets the design specifications. MATLAB Test Bench is used in conjunction with HDL Coder to infer fixed-point data types for floating-point to fixed-point conversion and generate test data for verifying the generated code.

Techniques for Test Case Generation

In addition to the tools provided by MATLAB, there are various techniques available for automated test case generation based on source code. These techniques aim to achieve different testing objectives and criteria.

Search-Based Test Data Generation

Search-based test data generation techniques have been widely investigated for code-level testing. These techniques target the testing of functional and non-functional properties and fulfill certain structural coverage criteria. While most of the research in this area focuses on code-level testing, there have been efforts to apply search-based techniques to higher levels of abstraction, such as MATLAB/Simulink models. These approaches provide a framework for test data generation and management, including the generation of mutant-killing test data.

Model-Based Test Generation

Model-based test generation is another approach used for automated test case generation in MATLAB. This approach involves creating test cases based on the components in the model. Test cases are generated for blocks and test harnesses in the model, and iterations are created for different scenarios. Model-based test generation allows developers to automatically create a set of test cases and iterations that correspond to the model elements. This approach is particularly useful for achieving model coverage and verifying the behavior of the model components.

Benefits of Automated Test Generation

Automated test generation based on source code offers several benefits for MATLAB developers and software development teams.

Improved Code Coverage

Automated test generation helps improve code coverage by generating test cases that exercise different parts of the code. It ensures that all branches, conditions, and statements in the code are tested, leading to better code quality and reliability.

Faster Test Execution

Automated test generation reduces the time and effort required for manual test case creation. It generates a large number of test cases automatically, allowing developers to focus on other critical tasks. This results in faster test execution and quicker feedback on the code's behavior.

Increased Test Effectiveness

Automated test generation techniques, such as search-based test data generation, can generate test cases that achieve higher coverage criteria and reveal potential issues in the code. These techniques explore different paths and scenarios in the code, leading to more effective testing and better detection of defects.

Test Case Reusability

Automated test generation produces a set of test cases that can be reused for regression testing. Regression testing ensures that changes or modifications to the code do not introduce new defects or break existing functionality. By reusing the generated test cases, developers can quickly validate the code after making changes, saving time and effort.

Conclusion

Automated test generation based on source code is a valuable technique for MATLAB developers to ensure the correctness and reliability of their code. MATLAB provides several tools, such as Simulink Design Verifier and MATLAB Test Bench, for generating test cases based on source code. Additionally, techniques like search-based test data generation and model-based test generation offer further options for automated test case generation. These techniques help improve code coverage, reduce test execution time, increase test effectiveness, and enable test case reusability. By leveraging automated test generation, MATLAB developers can enhance the quality and reliability of their software.

References

1. MathWorks. (n.d.). Test Case Generation. Retrieved from https://www.mathworks.com/help/sldv/test-case-generation.html
2. MathWorks. (n.d.). Source Code Generation and Verification. Retrieved from https://www.mathworks.com/help/ecoder/source-code-generation-and-verification.html
3. EcoTron. (n.d.). EcoCoder-AI - Automatic Code Generation Tool. Retrieved from https://ecotron.ai/ecocoder-ai/
4. GitHub. (n.d.). Test Generation. Retrieved from https://github.com/topics/test-generation?l=matlab
5. Springer. (n.d.). Automatic generation of system test cases from use case specifications. Retrieved from https://link.springer.com/article/10.1007/s10270-021-00924-8
6. MathWorks. (n.d.). MATLAB for Data Analysis. Retrieved from https://www.mathworks.com/products/matlab/data-analysis.html
7. MathWorks. (n.d.). Generate Test Cases for Model Decision Coverage. Retrieved from https://www.mathworks.com/help/sldv/ug/generate-test-cases-for-model-decision-coverage.html
8. MathWorks. (n.d.). Generate Test Cases from Model Components. Retrieved from https://www.mathworks.com/help/sltest/ug/generate-test-cases-from-model-components.html
9. MathWorks. (n.d.). MATLAB Test Bench Requirements and Best Practices for Code Generation. Retrieved from https://www.mathworks.com/help/hdlcoder/ug/matlab-test-bench-best-practices.html
10. MathWorks. (n.d.). Requirements Modeling and Automated Requirements-Based Test Generation. Retrieved from https://www.mathworks.com/company/newsletters/articles/requirements-modeling-and-automated-requirements-based-test-generation.html
11. ScienceDirect. (n.d.). A search-based framework for automatic testing of MATLAB/Simulink models. Retrieved from https://www.sciencedirect.com/science/article/pii/S016412120700129X