Model-based Automated Testing using a Record-Replay Mechanism

Junwon Kang¹, Hyunmin Yoon², Jaemoon Seo¹, Jungwon Roh¹, and Minsoo Ryu¹
¹Department of Computer Science and Engineering, Hanyang University, Seoul, Korea
²Department of Electronics Computer Engineering, Hanyang University, Seoul, Korea

Abstract – Automated testing is increasingly becoming an essential part of software development as it significantly reduces laborious and time-consuming manual efforts. By automatically constructing and executing test cases with minimal manual work, automated testing promises high productivity, better coverage, and reduced cost. However, there still exist many obstacles that must be addressed to successfully incorporate automated testing into a real software development process. In this paper, we present a novel approach to automated test case creation and execution, called MoReT (Model-based Replay Testing), for event-driven systems. MoReT accepts a finite state machine model for a target system under test and generates a series of test cases based on the notion of Chow’s n-switch coverage. Specifically, for a fixed n, MoReT finds an n-switch set cover that covers every sequence of consecutive transitions of length n+1 in a given state transition diagram. MoReT also allows us to automatically execute the generated test cases using a deterministic replay mechanism, called RT-Replayer, that can automatically generate and execute test cases, i.e., event sequences, during a test execution phase.

Keywords: Model-based testing, automated testing, deterministic replay mechanism, event-driven systems, finite state machine.

1 Introduction

Automated testing is increasingly becoming an essential part of software development as it significantly reduces laborious and time-consuming manual efforts. By automatically constructing and executing test cases with minimal manual work, automated testing promises high productivity, better coverage, and reduced cost. However, there still exist many obstacles, such as lack of adequate tool support and underestimation of the automation cost, that must be addressed to successfully incorporate automated testing into a real software development process.

One of key problems is that it is hard to automatically construct effective test cases for event-driven systems. Event-driven systems usually have various sources of external events including the user and a number of I/O devices such as sensors and network interface cards. As a result, there would be a huge number of possible permutations of events that will lead to different system responses. This makes exhaustive testing infeasible, and thus requires us to determine effective test cases that can satisfy appropriate test coverage criteria.

Another important problem is that it is hard to automatically execute test cases against an event-driven system. To test an event-driven system, we should be able to precisely generate desired I/O events such as sensor data acquisitions and network data arrivals during a test execution. This requires us to build a sophisticated test environment setting so that we can precisely control the generation of various I/O events. Unfortunately, this task could be overly difficult to manage because of the diversity and complexity of event sources and complicated interdependency between events.

In this paper, we present a novel automated testing approach, called MoReT (Model-based Replay Testing), for event-driven systems. MoReT addresses the above problems by providing a model-based test case creation method and an automated test case execution technique based on deterministic execution replay. First, MoReT provides a simple, efficient model-based method for automatic test case generation. It assumes a finite state machine model for a target system under test and generates a series of test cases based on the notion of Chow’s n-switch coverage [1]. Specifically, for a fixed n, MoReT finds an n-switch set cover that covers every sequence of consecutive transitions of length n+1 in a given state transition diagram. Second, MoReT provides a deterministic replay mechanism for automated test case execution. Specifically, MoReT can precisely generate and execute test cases, i.e., event sequences, during a test execution phase. MoReT executes those events automatically in pure software, completely obviating the need for human intervention and any external environment arrangement. Therefore, MoReT allows us to achieve automated test case execution for event-driven systems.
2 Automated Test Case Creation

For model-based test case creation, MoReT uses a FSM (finite state machine) model that can serve as a useful representation of an event-driven system where events govern the system’s operation and dynamic behavior. For example, a sequence of events that trigger a particular execution path in a FSM model can be viewed as a meaningful test case against the target event-driven system.

Given a FSM model, MoReT automatically generates event sequences as test cases against the target system. Since the number of event sequences is often unbounded in many cases, MoReT uses the Chow’s n-switch coverage to guide the test case selection. For a fixed n, an n-switch cover set consists of every sequence of consecutive transitions of length n+1 in the FSM diagram. For example, a 0-switch cover set consists of every possible state transition and a 1-switch cover set consists of every sequence of two consecutive transitions. Thus, we can easily control the test coverage by specifying the value of n.

MoReT first generates the 0-switch cover set by transforming a given FSM model into the 0-switch cover tree that contains every possible state transition of the FSM. Figure 1 shows an example FSM model and its 0-switch cover tree. In this example, the 0-switch cover set is the set of all the paths from the root node to leaf nodes.

Based on the 0-switch cover tree, MoReT can generate the n-switch coverage set. As mentioned earlier, an “n-switch” is a sequence of consecutive branches of length n + 1. Thus, the n-switch cover set is the set of all the paths from the root node to leaf nodes in the n-switch cover tree. Fortunately, we can obtain an n-switch cover tree by adding one more possible transition to every leaf node of (n-1)-switch. Thus, starting from the 0-switch cover tree, we can find the n-switch cover tree via iterative tree expansion.

![Figure 1. FSM model and its 0-switch cover tree.](image)

3 Automated Test Case Execution

Once test cases, i.e., event sequences, have been obtained, MoReT allows us to automatically execute the test cases using its replay mechanism, called RT-Replayer. RT-Replayer is a software mechanism that provides two key functions, event logging and event replay. For event logging, RT-Replayer monitors every I/O event and records them in an event log. When the kernel terminates, RT-Replayer stores the event log in safe non-volatile storage. For event recording, RT-Replayer can precisely emulate the recorded events consulting the event log during a re-execution.

![Figure 2. Manual test recording with a 0-switch cover set.](image)

Note that since a FSM model describes the target system at a higher level of abstraction, the event sequences derived from the logical FSM model cannot be directly executed against the physical target system due to the difference of abstraction levels between the FSM model and the physical system.

MoReT addresses this problem by introducing a two-step test execution process. The first step is manual test recording. In this step, we manually test the target system traversing every possible state transition, i.e., the 0-switch cover set, in the given FSM. In doing so, we use the event logging mechanism of RT-Replayer to record physical events and match them with the corresponding abstract events. The
section step is automated test execution. In this step, we can automatically execute the \( n \)-switch cover set using the event replay mechanism of RT-Replayer. RT-Replayer can automatically emulate all physical I/O events in pure software, completely obviating the need for human intervention and any external environment arrangement.

4 Conclusion

We have presented a novel approach to automated test case creation and execution, called MoReT (Model-based Replay Testing), for event-driven systems. The contributions of this work are two-fold. First, MoReT allows us to automatically create test cases from a FSM model for a target event-driven system. Second, MoReT provides a deterministic replay mechanism for automated test case execution.

5 Acknowledgment

This work was supported partly by Seoul Creative Human Development Program (HM120006), partly by Mid-career Researcher Program through NRF (National Research Foundation) grant funded by the MEST (Ministry of Education, Science and Technology) (NRF-2011-0015997), partly by the IT R&D Program of MKE/KEIT [10035708, “The Development of CPS (Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software”], and partly by the MSIP(Ministry of Science, ICT&Future Planning), Korea, under the CITRC (Convergence Information Technology Research Center) support program (NIPA-2013-H0401-13-1008) supervised by the NIPA (National IT Industry Promotion Agency).

6 References


