Extended Finite State Machines-Based Testing Using Metaheuristic Search-Based Techniques: Issues, and Open Challenges

Aneesa Saeed  
Department of Software Engineering  
Faculty of Computer Science and IT, University of Malaya  
Kuala Lumpur, Malaysia  
Email: aneesa.saeed@um.edu.my

Siti Hafizah Ab Hamid  
Department of Software Engineering  
Faculty of Computer Science and IT, University of Malaya  
Kuala Lumpur, Malaysia  
Email:siti.hafizah@um.edu.my

Abstract—Extended finite state machines (EFSM) models are currently widely utilized to model embedded and control systems. Thus, there is recent upsurge studies to generate test data from EFSM models (EFSM-based testing). Applying metaheuristic search-based techniques (MHSBT) for automating the process of test case generation from EFSM models became a proliferated field of interest during the last decade due to MHSBT find the optimal set of test cases among all possible test cases at reasonable cost. However, successful futuristic MHSBT for EFSM-based testing demand deep insight into the existing solutions that underlines stringent issues and challenges, which are lacking. The objective of this study is to analyze the current state-of-the-art of the application of MHSBT for EFSM-based testing. The study investigates the main issues in EFSM-based testing, including cost, continuous data, infeasible path, complex data structure, and concurrency. The current applications of MHSBT to solve these issues were elucidated. This study advocates that the majority of problems stem from the intrinsic features of EFSM models. Several open issues on EFSM-based testing adoption are presented as future research directions.

keywords: Software Testing; Extended Finite State Machines; Model-Based Testing; Metaheuristic Search-Based Techniques; Test Case Generation

I. INTRODUCTION

Extended finite state machines (EFSM) are currently widely utilized to model embedded and control systems. Since, 40 billion embedded systems are expected to be in operation by 2020 [1]. EFSM model is employed as the underlying behavioral model in various modeling languages such as SDL, UML, and Simulink/Statechart. EFSM models distinct states and their transitions; the transition of EFSM models contains predicate (guards condition/precondition), and operation (assignment/postcondition). An event will be received by the system that triggers the transition to another state. Therefore, they can specify both control and data parts in the system.

Extended finite state machines (EFSM)-based testing aims to generate executable test cases by systematically analyzing the EFSM models of the software system by following a test strategy (such as a coverage criterion which aims to cover specific features of the model). EFSM-based testing, is a part of model-based testing (MBT), has been proliferated field of interest recently. EFSM-based testing is a non-trivial task due to EFSM and their alternatives include various sophisticated constructs facilitating the definition and the triggering of the guards conditions. Many strategies have been applied for generating test case from EFSM models such as model checking [2], symbolic execution [3], and metaheuristic search-based techniques (MHSBT) [4]. The latest endeavor is to deploy MHSBT to EFSM-based testing which has been a field of interest recently. One reasons, among the bevy of inducements, the process of test case generation can be formatted as an optimization problem. MHSBT utilize heuristics to get solutions for a certain optimization problems at a affordable computational cost. Second reason, EFSM has been an interest modeling approach for both industry and academia compared to MHSBT which gained less attention in industry. Moreover, the ability of MHSBT to find the optimal set of test cases among all possible test cases at reasonable cost.

From the literature, many available studies reviewed various perspectives in model-based testing (MBT) [5]–[10]. However, only little studies that reviewed the applications of using state-based models (FSM, EFSM) in MBT [6], [9]. On the other hand, comprehensive studies [11]–[14] have reviewed MHSBT for structural testing purpose based on software code. All these surveys provided a good overview of the current state-of-the-art of each MHSBT and MBT separately, however, none of them can provide comprehensive inspections of the existing research in the area of MHSBT for EFSM-based testing. To the extent of our knowledge, analyzing of EFSM-based testing issues in the MHSBT domain is a nascent literature and requires comprehensive study and analysis. The objective of this study is to comprehensively analyze the current state-of-the-art of the application of MHSBT for EFSM-based testing. We also investigate the current issues, and solutions to present the taxonomy. Several open challenges are presented to direct the future directions.

The rest of this paper is organized as follows: Section II presents the related background to our focus. Section III presents and discusses the investigated issues, and current solutions. Open issues that can direct future trends are described in section IV. Conclusion and future work are presented in section V.
In the following, we present a brief formal notion about EFSM models. EFSM models are extension from the traditional Mealy Finite State Machine (FSM) with state variables, predicate (guards condition/precondition), actions (assignment/postcondition), and parametrized inputs and outputs. An EFSM M is defined by the tuple

$$M = (S, s_0, E, T, Var)$$

where $S$ is a set of states and should not be empty where each state has: a set of inputs $X$, a set of output $Y$, and may $X$ has a set of associated parameters $P$. $s_0$ is the initial state, $s_0 \in S$ $E$ is a set of events $T$ is a set of transitions. A transition $t \in T$ has source state $so(t) \in S$, a targeted state $tar(t) \in S$, and label $l(t)$. The label $l(t)$ is in the form $e[g]/a$, where $e \in E$, $g$ is a guard condition(s), and $a$ is a sequence of actions, called predicates. An action can be either internal to the machine or a shared action. , and $Var$ is a set of context variables.

Fig. 1 depicts a real example from EFSM models taken from study [25]. The example represented the three states model with six transition. From the Fig. 1, the states are $S = S_1, S_2, S_3$ with initial state $s_0 = S_1$, with states input $X = aa, ab, ba, bb, b$, and with output $Y = 00, 01, 11, 10, 1, 0$. The context variables are $Var = v_1, v_2, v_3$ and parameters $P = p_1, p_2$ are both used in formulating the set of events $E = p_1 \geq 10, p_1 \leq 20, p_2 \leq 0, p_2 \leq 10$ and $t_1$. Six transition in the model are $t_1, t_2, t_3, t_4, t_5, t_6$. Example for how the model work in the following: $M$ is at state $S_1$ then in order to trigger $t_1$, an input $aa$ is needed together with two input parameters $p_1$ and $p_2$. If the values of $p_1$ and $p_2$ satisfy the guard of $t_1(t_1(p_1 \geq 10, p_1 \leq 20, p_2 \geq 0, p_2 \leq 10)$ then $t_1$ is triggered, the operations $(v_1 := p_1; v_2 := p_2)$ are executed and the machine outputs 00. These operations upgrade the values of $v_1$ and $v_2$. Since $t_1$ ends at the same state, $S_1$, then the machine remains at this state.

### III. EFSM-based Testing Issues

Several problems encumber EFSM-based testing adoption and success. In this section, we discuss the EFSM-based testing problems and review the existing efforts based on MHSBT that aim to alleviate these problems. Table I summarizes the issues and the current solutions.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Current Solutions</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infeasible path</td>
<td>Estimating approach</td>
<td>[26]-[28]</td>
</tr>
<tr>
<td></td>
<td>Static analysis</td>
<td>[25], [29], [30]</td>
</tr>
<tr>
<td></td>
<td>Model execution</td>
<td>[31], [32]</td>
</tr>
<tr>
<td></td>
<td>Reachability analysis</td>
<td>[33]</td>
</tr>
<tr>
<td>Complex Constraint</td>
<td>MHSBT-based constraint solver</td>
<td>[4], [21], [34]</td>
</tr>
<tr>
<td>Continuous Data</td>
<td>Approximation approach</td>
<td>[35]</td>
</tr>
<tr>
<td></td>
<td>Model instrumenting</td>
<td>[36]</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Feedback approach</td>
<td>[37]</td>
</tr>
<tr>
<td>Cost</td>
<td>Similarity measuring</td>
<td>[38]</td>
</tr>
</tbody>
</table>

### A. Infeasible paths

The crucible problem in EFSM-based testing that addressed intensively in the literature is that generating test data from EFSM models whose transitions path is not feasible, in which there are conflict guards in the path transitions. An example of the infeasible path occur when one transitions operation may
assign the value 10 to a variable x while a advance transitions guard needs \( x \geq 10 \) in spite of the value of x not having modified between these transitions. Therefore generating test data that trigger these infeasible paths is time consuming. In the literature, the research attempted to detect only feasible paths and generate test data for triggering these feasible paths. The main directions for applying MHSBT to solve infeasible path problem are described in Table II.

From Table I and II., we conclude the proposed approaches to solve this problem are categorized into four approaches. The first type (estimating approach), proposed in [26]–[28], depended on estimating the feasibility of transition paths and generating transition paths which are more probably to be feasible. Derderian K. et. al [27] proposed fitness function to estimate the feasibility of the path based on the ranking the penalty values of conditions of transitions paths, in which lower fitness metric value means the probability of path feasibility is more. The approach in [26] proposed GA with extending the transition feasibility metric presented in [27] by categorizing the state transitions into two types: affecting and affected-by. Moreover, they assigned penalty value to a certain path based on the assignment type, condition’s guard type and the operator. In [28], the authors extend the penalty estimating approach as presented in [27]by deeming not only the transition guards conditions (constraints) but also time temporal constraint. This approaches in finding infeasible paths relied on the penalty values which are investigated by the authors which are limited to be generalized.

The second type (static analysis), presented in [25], [29], [30], focused on measuring the data-flow dependency in the model which are statistical analysis based as well as penalty estimating approach. Study [29] extended the previous transition feasibility metric in [26] by measuring the data-flow dependencies to handle counter variables which means the variables that counts how many times a transition is repeated. Furthermore, the same authors in another study [25] presented full empirical study on the proposed metric and generating data that triggers the generated paths. The test case generation approach by Kalaji et al. [25], [29] requires two stages: one to generate feasible transition paths, and other one to generate test data to trigger the feasible paths. In other study [30], transition feasibility metric in [26] was extended by distinguishing the guards condition into hard guards and easy guards. This studies further are limited to be generalized because of they depended on the previous type (penalty estimating approach).

The third type (model execution) applied by the same authors in two studies [31], [32], premised on converting EFSM model to executable model which the feasible paths only will be generated. In both studies, a multi-objective EA is proposed, which considers two objectives: to search for a test sequence that covers a target transition, as well as to minimize the length of this test sequence. The main difference between the two studies is the model analyzer component that uses dependence analysis in [32], instead of slicing of the model in [31].

The fourth type is applied on [33] based on reachability analysis, in which they used breath-first search to find paths where no condition conflicts exist. An empirical study on the efficiency of MHSBT for EFSM is also provided and one of their findings is a positive correlation between the cost of test case generation and some other metrics, such as the number of numerical equal operators in the path conditions.

Table II shows the solution of the sub-issues that affect the infeasible path detection. Losing information in detecting the infeasible path is the critical problem faced when searching deals with state variables. Most of the studies deal with this problem by applying dependency analysis because the lack of context variables controllability in the EFSM models by the MHSBT. Another sub-issues affect the successful of infeasible path detection is that whether the EFSM models are contains nested loops. Infeasible path detection method may not successfully find out the values for deeply nested loops. This issue covered by only four studies which need further investigations, providing the correlation factor is an important aspect in evaluating the performance of infeasible path detection methods. All the reviewed studies provided this analysis in the evaluation.

B. Generating continuous test data

The creation of continuous input data by numbers arrays, is difficult to achieve for real-world models while these usually need the data to be in a certain minimum length. The existing research to solve this problem are conducted by Windisch. Andreas [35], [36]. In [35], Windisch, Andreas investigated EA to generate continuous input data(signal) for real-time simulink/stateflow models: the signals were generated by a sequence of individual signal blocks based on approximation. Moreover, Windisch, Andreas extended the previous work to generate a valuable, realistic continuous signals in [36]. In [36], the proposed method includes both instrumenting statecharts for tracing resulting control flows and calculating fitness values using specific distance measures for statecharts. Distance metrics, based on approach level with normalized branch level, are used to assess the test datas ability to solve conditional expression, thus guiding the evolutionary search. Industrial case study was used to evaluate the performance of the proposed method.

C. Complex Constraint

The prominence industrial needs move toward a software with growing complexity and demanding quality. Test data generation for complex industrial systems is a nontrivial problem as contains many sophisticated constructs facilitating the definition of constraints. These complex systems include complex data structure and constraint. There are existing research that attempts to develop scalable, and applicable techniques that handle the complex features of these kind of systems. In [4], [21], [34], the authors proposed (1+1) EV, GA, and variable techniques to generate test data from UML state machines that solve Object Constraint Language (OCL) constraints. They developed a constraint solver based on GA and (1+1) EV to generate test data for satisfy transition guards constraint in complex industrial systems. Therefore, they extended the general form of fitness function by defining branch distance functions for various types of expressions in OCL (such as enumerations and collections) to guide search algorithms. The work of Shaukat et al. focused on transition’s constraint but they did not consider this transition constraint may affect the feasibility of a particular path contains this transition. Vos et. al [39] presented empirical study for investigating the
scalability of the evolutionary functional framework within specific automotive industrial setting through two case studies. One of his findings is it is necessary to have a certain level of evolutionary computation skills and be able to devote a significant amount of time to define and refine a suitable fitness function. This challenge is compensated by the results in some cases but it is still an significant instance encumbers the total acceptability of MHSBT for functional testing in industry.

D. Concurrency

One of the challenge in generating test data is the EFSM models contain hierarchical and/or a concurrent structure defined over state. To the best of our knowledge, the existing solution based on MHSBT to handle this problem was conducted by Oh, Jungsup et. al [37] based on feedback approach. They proposed messy-GA based framework to generate transition coverage adequate input sequences without fixing the length of sequences in advance. The approach is based on using the feedback from the executable model. It is able to cope with cyclic paths and concurrency. The framework consists of three main components; the executable model generator, the coverage goal generator and the test case generation.

E. Cost

Applying MHSBT for EFSM-based testing usually generate large number of test cases when applied to real-world systems, whatever the coverage criteria. The cost of executing and evaluating these test cases is expensive. Therefore, there is a need to choose a small enough set of these generated test cases that have the highest potential fault detection power. The existing solution to select subset of the test cases and reduce the cost is by measuring the similarity between the generated test cases as proposed by [38]. They compared different similarity measures in terms of what test case information should be assess (test case encodings) and how they conduct this assessment (similarity functions).

IV. OPEN CHALLENGES

From the literature, applying one of MHSBT for EFSM-based testing poses a number of non-trivial problems which still need further investigations as described below:

A. Infeasible paths

EFSM models are used to model large systems such as embedded and control systems; these kind of systems may have infeasible paths. These infeasible paths is exist in the models because of the variable correlation among the actions and predicate conditions. Generating test cases from these infeasible paths are time-consuming. However, deciding the feasibility of a specific path is unspecified thus the development of novel techniques is still an open issue.

B. Hierarchical and concurrent models

Complex systems are usually developed gradually in many steps during a process of improvement. In these cases, Instead of generating test cases straightly from the final system may not be practical, generating test cases from the unrefined version of the system. The final test cases then will be developed in parallel to the refinement of the system models. In addition to, large complex real-world systems can be divided into simpler components. Similarly, there is a need for testing method to integrate test cases of the components into the test cases of the whole system. The current testing methods for hierarchical and concurrent FSM models could be utilized as a starting point for proposing sufficient testing methods for EFSM models.

C. Complex data types

Very often the EFSM models involves relatively simple data structures, such as numeric or enumeration types, that could be faciley converted onto numeric. However, complex data types like sets, collections, or partial functions require additional investigations. Therefore, applying MHSBT for EFSM models with complex data structures needs efficient chromosomes encodings approaches and effective fitness functions that solve the constraints. In addition to, a necessity for novel trust MHSBT similar to [4] exists for other modeling and constraint language and which are required to be low cost and efficient.

D. Cost

The factor that significantly increment the cost of EFSM-based testing are the high cost of test case execution. Therefore, the number of the test cases must be as small as possible with extent possible preserving their fault detection power. In real-world EFSM models, MHSBT usually generate large number of test cases whatever the coverage criteria. Therefore, the current approaches to reduce the cost is by selecting an optimal set of the generated test cases to be executed. These solutions requires two approaches: one for generating the test cases, and the other one for selecting. However, a need for developing novel trust techniques which are required to generate optimal set of test cases.

V. CONCLUSIONS

This paper reviewed the crucial intrinsic restrictions of issues in EFSM-based testing and the existing solutions based
on MHSBT to motivate for more emergence of effective and efficient MHSBT for EFSM-based testing. We have sketched some of the main features of EFSM models that cause the issues in EFSM-based testing. We also performed a literature review of the techniques that proposed for solving the identified issues. The existing challenges of applying MHSBT for EFSM-based testing have been outlined. We believe that this review will assist to advance applying MHSBT for EFSM-based testing, since this area requires additional efforts to address a number of crucial EFSM-based open challenges, which expect as future research directions in this area.

ACKNOWLEDGMENT

The authors acknowledge the support provided by University of Malaya Research Grant: RG 106-12ICT, “A language for Program Analysis of Java in Agile Development Environment.”

VI. REFERENCES

REFERENCES


