Generating Feasible Input Sequences for Extended Finite State Machines (EFSMs) using Genetic Algorithms

Karnig Derderian
Brunel University
Department of Information Systems and Computing
Uxbridge UB8 3PH, UK
karnig.derderian@brunel.ac.uk

Robert M. Hierons
Brunel University
Department of Information Systems and Computing
Uxbridge UB8 3PH, UK
rob.hierons@brunel.ac.uk

Mark Harman
Department of Computer Science
King's College London
London, WC2R 2LS, UK
mark@dcs.kcl.ac.uk

Qiang Guo
Brunel University
Department of Information Systems and Computing
Uxbridge UB8 3PH, UK
qiang.guo@brunel.ac.uk

Categories and Subject Descriptors

D.2.5 [**Testing and Debugging**]: Testing Tools

General Terms

Algorithms, Theory

Keywords

EFSM, GA, Input Sequences, Transition Path

1. INTRODUCTION

Testing is an important part of the software engineering process but can be time consuming, error-prone and expensive. Test automation can help reduce these problems.

Many state based systems, like protocols, have been modelled as finite state machines (FSMs) and extended finite state machines (EFSMs). They have been an effective method of modelling because a variety of techniques and automated tools exist that work with them. To ensure the reliability of these systems once implemented they must be tested for conformance to their specification.

Usually the implementation of a system specified by an FSM or EFSM is tested for conformance by applying a sequence of inputs and verifying that the corresponding sequence of outputs is that which is expected. This commonly involves executing a number of transition paths, until all transitions have been tested at least once. In EFSMs the feasibility of a transition path depends on satisfying all the transition guards involved, in addition to finding a specific input sequence to trigger these transitions.

This poster addresses the issue of finding feasible transition paths and generating input sequences for systems based on the EFSM model. A novel way of abstracting parts of the data in the EFSM in order to facilitate the generation of a feasible transition paths using GA is presented.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

2. PROBLEM AND SOLUTION

As EFSMs we refer to those Mealy (finite state) machines with parameterised input and output, internal variables, operations and predicates defined over internal variables and input parameters. The label of a transition t in an EFSM has two guards that decide the feasibility of the t: the input guard g_I and the domain guard g_D. In order for t to be executed g_I, the guard for inputs from the environment must be satisfied. Some inputs may carry values or specific input parameters and g_I may guard those values with the input parameter guard g_P. Note that in order to satisfy the domain guard g_D of t, it might be necessary to have taken some specific path to the start state of t.

In EFSMs a transition path depends on the result of the input parameter guard and the domain guard. The result of these guards is determined dynamically depending on the values of the internal variables and input declarations, which in turn can assume different values after each transition. Hence some transition paths might have no conditions associated, some might have conditions that are difficult to satisfy and some transition paths will be infeasible. The existence of such infeasible transition paths creates difficulties for automating the test generation process for EFSMs. The machines that arise are complex and brute force exponential testing is infeasible [4]. Automating the test sequences generation for EFSMs has been of interest [4, 5, 6, 1].

One way of approaching the problem is to abstract away the data part of the EFSM and consider it as an FSM on its own. However a transition sequence for the underlying FSM of an EFSM cannot be guaranteed to be feasible for the actual EFSM. Another way is to expand an EFSM to an FSM and then use the techniques used for FSMs. However this can lead to a combinatorial explosion.

Some recent work on generating feasible conformance test sequences for EFSMs was presented in [1]. Our EFSM model

is more general as we do not consider only linear constraints. [1] gives a comprehensive summery of all the important work in this field up to now. Test generation for EFSMs is still an open research problem [1]

Generating a feasible transition path and a corresponding input sequence for EFSMs can help with test sequence generation for EFSMs. The problem of finding a (an arbitrary) feasible transition sequence for an EFSM and generating the necessary input sequence to trigger, depending on the variables and conditions involved, is generally uncomputable.

Definition 1. A feasible transition path (FTP) from state s_i to state s_j of an EFSM M is a sequence of transitions initiating from s_i and ending at s_j that are feasible for at least one combination of values of the finite set of internal variables V of M.

Not all transitions in EFSMs have input parameter guards and domain guards and so transitions in an EFSM M can be categorised in the following four types: simple transitions are those transitions that have *no* guard; IPB transitions are those transitions that *have* input parameter guard but *not* a domain guard; DPB transitions are those transitions that *have* a domain guard but *not* an input parameter guard; and IPB-DPB transitions are those transitions that *have* both an input parameter guard and a domain guard.

Some transition paths consist of transitions with difficult to satisfy guards. The presence of simple transitions in a transition path makes it more likely to be an FTP since there are no guards to be satisfied. While a random algorithm could be used it does not always produce acceptable results. In [2] GAs were used to generate UIOs for FSMs more efficiently than a random algorithm can.

The objective of this work is to facilitate the generation of FTPs in EFSMs. Then the necessary input to trigger them can be generated. We focus on generating transition paths that are likely to be feasible. The overall approach is defining a fitness function that estimates how likely is that a transition path is feasible and how easy is it to generate an input sequence to trigger it. A GA is used in attempt to generate such transition paths. The genotypes with highest fitness are verified and GA execution repeated if necessary.

3. RESULTS AND CONCLUSIONS

A set of experiments were conducted on the EFSM M representing the Inres protocol example [3]. A GA was used in attempt to generate FTP between the initial and final states of M. The results were compared to randomly generated transition paths. To measure the feasibility of each potential FTP a random input parameter generator was used. It randomly generates 1000 potential input parameters within a specified range. FTPs where the random input parameter generator found more input parameters that make the transition path feasible were considered to be better.

Figure 3 shows some initial results of generating FTPs for M. These results represent the best of 50 attempts for each FTP length. FTPs of lengths 3 to 10 were considered since the shortest path between the two states is 3 transitions and the search became more difficult as the length was increased.

The GA and random generation algorithm were given at least the same amount of generation and feasibility test effort in terms of the random input parameter generator. The random algorithm could not find any FTPs while the GA found one for each length. The random generation algorithm was given extra effort for each FTP length until it generated an FTP. Hence the graph representing the fitness evaluations it took the random algorithm to generate a result for each length is steeper that the one plotted by the GA on Figure 3. The graph seems to indicate that as the problem with finding FTPs gets larger the GA's dominance margin in evaluations needed increases.

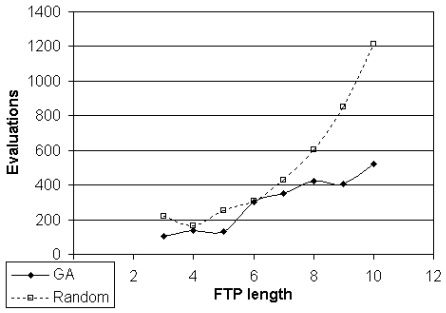

Figure 1: Number of evaluations to generate FTPs for the Inres example by GA and random generation

Although the experiment is small it indicates the limitations of the random algorithm generation and the potential of an EFSM abstraction method and fitness function. Generating FTPs using a fitness function heuristics like GAs can offer an advantage for automated testing of EFSMs.

We have define a novel EFSM abstraction that can help us represent the problem of finding an FTP and generating the input sequence needed to trigger it. The fitness function rewards transitions with more easy to satisfy guards in order to minimise the effort of input sequence generation. Some results indicate that the fitness function rewards FTPs with easy to satisfy conditions. The GA used also seems to outperform the random generation algorithm with an increasing margin, as the length of the transition path is increased.

4. REFERENCES

[1] A. Y. Duale and M. Ü. Uyar. A method enabling feasible conformance test sequence generation for EFSM models. *IEEE Transactions Computers*, 53(5):614–627, 2004.

[2] Q. Guo, R. M. Hierons, M. Harman, and K. Derderian. Computing unique input/output sequences using genetic algorithms. In *LNCS vol. 2931*, pages 169–184. Springer, 2004.

[3] R. M. Hierons, T. H. Kim, and H. Ural. On the testability of SDL specifications. *Computer Networks*, 44:681–700, 2003.

[4] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - a survey. *Proceedings of the IEEE*, 84:1090–1123, 1996.

[5] X. Li, T. Higashino, M. Higuchi, and K. Taniguchi. Automatic generation of extended UIO sequences for communication protocols in EFSM model. In *Distributed Processing System No.066*, pages 225–240, Japan, November 1994. IPSJ SIGNotes.

[6] A. Petrenko, S. Boroday, and R. Gorz. Confirming configurations in EFSMs. *IEEE Transactions on Software Engineering*, 30:29–42, January 2004.