A Two Steps Test Suite Generation Approach based on Extended Finite State Machines

Ana Ţurlea
turleaana@gmail.com

Department of Computer Science, University of Bucharest,
Bucharest, Romania

Abstract. Using extended finite state machines for test data generation can be a difficult process because we need to generate paths that are feasible and we also need to find input data that traverse a given path. This paper presents a two steps test generation algorithm for extended finite state machines. The first phase produces a set of feasible transition paths that cover all transitions and the second phase generates input sequences that trigger each path. The first step uses a modified multi-objective genetic algorithm (deleting redundant paths and shortening the paths) and the second phase uses a hybrid approach combining genetic algorithms with local search techniques. The multi-objective problem aims to optimize the transitions coverage and the paths feasibility, based on dataflow dependencies.

Keywords: genetic algorithm, extended finite state machines, multi objective genetic algorithm, hybrid genetic algorithm, local search, global search, transitions coverage, paths feasibility, test suite, test data generation

1 Introduction

Finite state machines (FSMs) are widely used in test data generation approaches. A FSM consists of a finite set of states and transitions between states. Each transition has a start state, and end state, an input and produces an output. An extended finite state machine (EFSM) extends the FSM with memory (context variables), guards for each transition and assignment operations. In FSMs all paths are feasible, but the existence of context variables in EFSMs can lead to infeasible paths. Using EFSMs in test data generation, we are dealing with feasibility problems. There are many coverage criteria used in EFSM testing. A set of inputs satisfies the transition coverage criterion if the execution of all test cases leads to all transitions being triggered. Our proposed approach is a two phases algorithm generating feasible paths based on transition coverage criterion and then generate input data for those paths. There are other similar methods, but our technique uses a modified multi-objective algorithm, having two objectives (transition coverage and path feasibility) and a solution shortening operator, for generating a test suite and a hybrid genetic algorithm for the test data generation.

2 Two steps algorithm for test data generation

Phase 1 - Multi-objective Algorithm: Paths generation: The first step uses a multi-objective algorithm and generates different collection of paths, taking into account data dependencies and transition coverage. We use two objectives, that may conflict.

Solution Encoding: a solution is a set of paths. A chromosome has variable length and is composed of genes (paths of fixed length). We put a limit on the total number of genes equal with the number of transitions (as we want to cover all transition we need a number of paths less or equal to the total number of transitions). To encode a path we adapt the encoding used by Kalaji et al. [2]. A path is represented by a sequence of integer values, each number defining a transition.

The Genetic Operators used are inspired from [1] and adapted to our problem. *The Mutation Operator* can change a chromosome in many ways, since we have different characteristics for the individuals. We will apply the following changes, with equal probability: add a random generated gene, replace a gene from a random point, remove a random selected gene, replace a value for a random gene from a random index, exchange materials between two random genes. *The Crossover Operator* creates two new chromosomes from the two existing parents chromosomes. Our algorithm applies two recombination methods with equal probability, as it follows: identify one gene in each parent at the same index (must be a valid index for both parents, since we have variable sized chromosomes) and interchange genes at this position; exchange materials between two genes from a valid random index.

Objective Functions: In this approach we use two objective functions, maximizing the path feasibility and transition coverage. To determine the feasibility of a path we adapt the metric described by Kalaji et al. in [2]. The feasibility metric guides the search towards transition paths that are likely to be feasible using dataflow dependencies among the transitions. The second objective function computes the transition coverage as follows: for each transition we count the occurrences in all paths and return this number if all transitions were covered or INF otherwise. In our experiments INF was equal to 10000.

The execution of the NSGAII-modified algorithm produces a population of solutions. Each solution represent a set of paths. After each evolution of the genetic algorithm we optimize each individual removing redundant paths and transitions as follows: sort the paths ascending according to the feasibility objective function; for each path we try to remove the last transitions if thy were covered already in the previous visited paths and mark as visited the remaining transitions. After modifying the chromosomes, we replace the current population and continue the evolution until the stopping condition is reached.

Phase 2: Hybrid Genetic Algorithm for Test Data Generation GAs tend to the global optimum, but may take a long time to converge to the optimal solution. To overcome this problem, in the second step of this method we use a Hybrid Genetic Algorithm (HGA) proposed in our previous work [4]. For each path generated at the first step, we generate input data that validates the guards and triggers all transitions. This method uses a genetic algorithm combined with a local search method.

The algorithm ends when the stop criteria is reached or when the maximum number of evolutions is exceeded. After the selection step, a new generation is created using recombination, crossover and mutation. After each evolution of the genetic algorithm, the best chromosome is selected to be improved locally. Applying local search we may find a better individual. If local search returns a chromosome with better fitness function this new one will be added to the population and it will be used in the next evolution. If there is no improvement, the algorithm continues with the next evolution.

In our previous work [4] we used a Alternate Variables Method, as it follows: for each chromosome selected by search algorithm, we take all genes in the reverse order. For each gene we start exploratory moves (+1 and 1 for integer variables) and decide in which direction to search for better values. A successful move consists in improving the fitness value. If a +1 move was successful, then, with each iteration i of the search, we add = 2 i to that gene. Otherwise, if the 1 move was successful, we subtract from the gene. We iterate the search as long as the new chromosome (with the modified gene) has a better fitness function. The fitness function used in this approach is: $fitness = approach\_level + normalized\_branch\_level$ ($f = al + nbl$). $branch\_level$ computes, for the predicate that is not satisfied, how close the predicate was to being true, using Tracey's objective functions [3]. The normalization function is $norm : [0, 101] \rightarrow [0, 1], norm(d) = 1 - 1.05^{-d}$. More details about the algorithm ca be found in [4].

3 Conclusion

In this paper we propose a two steps algorithm for test data generation for EFSMs using a customized multi-objective algorithm to generate a transition coverage test suite and a hybrid genetic algorithm to generate input data for each path. This approach is different from the existing methods, customizing the genetic operators, optimizing the solutions by deleting redundant paths and path transitions and by using the combination of genetic algorithms and local search methods. The second phase of the algorithm may take a long time to find a solution, especially for paths with grater length. To overcome this problem, we decided to shorten the paths, to find paths with smaller complexity and, in the same time, to cover all transitions. Experiments have shown that there is easy to generate input data for the sets of paths found at the first step of the algorithm.

References

1. Asoudeh, N., Labiche, Y.: Multi-objective construction of an entire adequate test suite for an EFSM. In: Proceedings of the 2014 IEEE 25th International Symposium on Software Reliability Engineering. pp. 288–299. ISSRE '14, IEEE Computer Society, Washington, DC, USA (2014), `http://dx.doi.org/10.1109/ISSRE.2014.14`

2. Kalaji, A.S., Hierons, R.M., Swift, S.: An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models. Information & Software Technology 53(12), 1297–1318 (2011)
3. Tracey, N.J., Clark, J.A., Mander, K., McDermid, J.A.: An automated framework for structural test-data generation. In: The Thirteenth IEEE Conference on Automated Software Engineering, ASE 1998, Honolulu, Hawaii, USA, October 13-16, 1998. pp. 285–288 (1998), https://doi.org/10.1109/ASE.1998.732680
4. Turlea, A., Ipate, F., Lefticaru, R.: A hybrid test generation approach based on extended finite state machines. In: Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2016 18th International Symposium on. pp. 173–180. IEEE (2016)