Empirical study on the efficiency of search based test generation for EFSM models

Ruilian Zhao*, Mark Harman† and Zheng Li†
*Department of Computer Science, Beijing University of Chemical Technology, Beijing 100029, China
†CREST, Department of Computer Science, King's College London, Strand, London, United Kingdom

***Abstract*—Experimental work in software testing has generally focused on evaluating the effectiveness and efficiency on various source code programs. However, an important issue of testing efficiency on the model level has not been sufficiently addressed, and hitherto, no empirical studies exist. This paper presents an automated test data generation system for feasible transition paths (FTP) on Extended Finite State Machines (EFSM) models and investigates the statistical properties of testing efficiency using statistical tests for correlation and formalisation according to the test data generated by applying the system on four widely used EFSM models. An important and encouraging finding is a close positive correlation between test generation cost and the number of numerical equal operators in conditions (NNEOC) on a FTP. In addition, as the NNEOC increases, there is a raising correlation between the test generation cost and the length of path with events variables (LPEV) or the number of numerical event variables on a path (NNEV), and NNEV increases linearly with the LPEV. Furthermore, empirical study shows that there is very strong exponential relationship between test generation cost and NNEV or LPEV only when NNEOC is considerable. The results provide a significant guide to predict the testing efficiency for EFSM models.**

***Keywords*-Search; Test generation; EFSM**

I. Introduction

Testing from formal specifications offers a simpler and more rigorous approach to the development of functional testing than standard testing techniques. Finite-state machines (FSMs) and extended finite-state machines (EFSMs) are among the most popular formal specifications. They are widely used in a number of industrially significant specification techniques, such as SDL, Estelle, Statecharts, UML, and so on. Although automated test sequence generation methods made substantial contributions toward test generation on systems specified as FSMs [6], [9], [13], [16], [21], [27], test generation for the EFSM models remains an open research problem. The difficulty of automating test generation for the EFSM models arises from the fact that, in general, an EFSM model contains infeasible paths due to the existence of the context variables. Moreover, finding a set of test data to trigger a given feasible path in an EFSM is a hard task, too [18]. Many techniques producing a set of paths for generating test sequences from an EFSM have already been reported in the literature [5], [7], [8], [10], [18]. However, there is comparatively little work on producing real test data for feasible paths in EFSMs.

On the other hand, experimental work in software testing has generally focused on comparing and evaluating the effectiveness and efficiency of different coverage criteria on various source code levels [3], [12], [15], [17], [19]. Gallagher et al. [14] reported the factors, *the number of test data variables being generated* and *the length of test path*, which affect the performance of the test data generator for Ada software system. However, the paucity of the efficiency analysis on test data generation at the model level of abstraction means that the software tester has little knowledge on potential factors that affecting the efficiency of test data generation in EFSM models.

Thus, this paper first aims to develop the infrastructure of automatic test data generation for EFSM models that produce real data to trigger feasible transition paths. Secondly, this paper provides empirical results on efficiency analysis of test data generation for a set of state-based models to address the following questions:

- Which factors affect the performance of test data generation in EFSM models?
- Which is a decisive factor?
- What correlation exists between the test generation efficiency and these factors?
- Which underlying regression model is better for predicting, linear or exponential?

The primary contributions of the paper are as follows:

1) The paper presents a genetic algorithm-based test data generation system for feasible transition paths in EFSM models, and empirically validates the efficiency of the system by applying the system to a set of EFSM models, with or without an EXIT state.
2) The paper also empirically confirms that the number of numerical equal operators in conditions (NNEOC) plays a key effect on the efficiency of test data generation. Furthermore, empirical study shows that there is a very strong linear correlation between the number of numerical event variables (NNEV) and the length of path with event variables (LPEV) on a feasible transition path, and the test generation cost grows exponentially along with the increasing of LPEV or NNEV when NNEOC is considerable.

The remainder of this paper is organized as follows. Section II defines two types of complete path, and introduces the main principle of test data generation algorithm using

GA on EFSM models. Section III briefly describes the test generation implementation, and gives some metrics used in test generation efficiency analysis. Section IV reports the experimental results and discussion. Section V reviews related work. Finally, conclusions and future work are given in Section VI.

II. Test data generation on EFSM model

One of the challenges with testing EFSM models is how to correctly account for path. This is because EFSM models can be non-terminating (i.e. without an EXIT state), which breaks traditional path used in testing.

In this section, we first introduce the syntax of EFSM models. Then we define complete transition path and potential feasible path which are used in the research reported in this paper. Finally, we describe how to generate test data using genetic algorithm for a feasible transition path from EFSM model.

A. Extended Finite State Model

An extended finite state machine (EFSM) is a 6-tuple (S, V, I, O, T, s_0), where S is a nonempty finite set of states, V is a nonempty set of internal/context variables, I is a nonempty set of input interactions, O is a nonempty set of output interactions, T is a nonempty set of transitions, and s_0 is an initial state [18]. Each member of I is expressed as $event(inputlist)$ meaning the interaction of event occurs with a list of input parameters *inputlist*, which is disjointed from V. Each member of O is expressed as $action(outlist)$ meaning action occurs with a formal list of parameters *outlist*. Each parameter in *outlist* can be replaced by a suitable variable from V, an input interaction parameter, or a constant. Each element t of T is a 5-tuple $t(source, target, input, condition, action)$. Here, $source$ and $target$ are the states in S representing the source state and the target state of t, respectively. $input$ is either an event from I or empty. $condition$ is a predicate as a set of logical expressions in terms of the variables in V, the parameters of the event and some constants. $action$ is a sequence of actions which consists of statements such as output statements and assignment statements and so on (we assume a standard expression language including assignments). All parts of a t are optional.

A state transition t occurs when one of the machine's transitions is taken. If a transition t has a condition c on the internal variables and input parameters, then c must be satisfied in order for t to be taken. A *self-looping* transition is a transition t where the source of t is the same as the target of t. A set of distinct transitions may have an identical source and an identical target. A *final transition* is one whose target is an EXIT state that has no outgoing transitions in a terminating EFSM model, or is one whose target is a START state in a non-terminating EFSM model.

B. Complete Paths in EFSM Model

A path is usually presented as a sequence of nodes or edges. By path of an EFSM we mean a sequence of adjacent transitions of an EFSM. Because EFSM can be non-terminating, this has led to following two types of complete path definitions.

Definition 1 (Complete transition path)**.** *A complete transition path is any path* $\pi = t_1 t_2 \cdots t_i \cdots t_n$ *that* $source(t_1)$*=START state,* $target(t_n)$*=EXIT state and* $target(t_i) = source(t_{i+1})$ $(1 \leq i < n)$ *in a terminating EFSM model.*

Definition 2 (K Complete transition path)**.** *A* K *complete transition path is any path* $\pi = t_1 t_2 \cdots t_i \cdots t_n$ *that* $source(t_1)$*=START state,* $target(t_n)$*=START, too and* $target(t_i) = source(t_{i+1})$ *in a non-terminating EFSM model, and there is* $K-1$ *transition* t_e *in path* π *whose* $target(t_e)$*=START state* $(1 \leq i, e < n)$.

Definition 3 (Condition conflict)**.** *A transition may not be traversed if there are conflicting conditions in the paths. A given path has a condition conflict if there exists a variable* v *and a pair of transition* (t_i, t_j)*, such that the current values of the variable* v *makes the condition of* t_i *to be* $True(False)$*, but results* $False(True)$ *in the condition of* t_j.

Definition 4 (Potential feasible path)**.** *A path that is free of condition conflict is called a potential feasible path.*

C. GA based Test Data Generation

Genetic Algorithms work with populations of candidate solutions to a problem. Our specific problem is to use a genetic algorithm to search a set of input data that can traverse a potential (K) complete FTP in EFSM models. More generally, given a particular (K) complete path π in an EFSM, $\pi = s_1 \xrightarrow{e_1[c_1]/a_1} s_2 \xrightarrow{e_2[c_2]/a_2} s_3 \cdots s_m \xrightarrow{e_m[c_m]/a_m} s_{m+1}$, where e_i is an event, c_i is a condition, and a_i is a sequence of actions, an individual is a list of input values, $x = (x_1, x_2, \cdots, x_n)$, corresponding to all parameters of the events $e_1, e_2, \cdots, e_m$ in the order they appear. If the sequence of events, having the parameter values $x_1, x_2, \cdots, x_n$, determines the transitions on the path π and validates the condition c_i of each transition, then x is a solution for path π. This means that each reached restrictions imposed by the path must be solved and previous predicates must remain solved, despite the changes on input values in the search process. Figure 1 shows a simple transition path with three transitions. The test input is the sequence of $(e1(a, b), e2, e3)$, where the variable a and b are replaced by real values. For example, $(e1(2, 3), e2, e3)$ is a validated test data that can traverse the path while $(e1(-1, 3), e2, e3)$ is not, as the condition on T2 is failed. Therefore, $(2, 3)$ and $(-1, 3)$ are individuals but only $(2, 3)$ is a solution for this example.

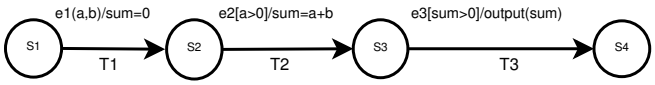

Figure 1. Example of test data

The genetic algorithm evaluates each individual by executing each transition on a potentially complete FTP with the values encoded in the chromosome's genes. A *fitness function*, assigning a score (fitness) to each chromosome in the current population, is used to compare the individuals and to differentiate their performance in each population. The fitter individuals are the ones which follow more transitions from the given path.

III. Experimental Setup

The experimental approach is straightforward. Firstly, potentially complete feasible paths with different lengths, varying from 3 to 50 depending on EFSM models, are produced by employing Breadth-First search. Secondly, for each path length, 5 paths are picked up to develop test data. For each path, ten test cases are generated by applying the GA. Finally, the test generation efficiency is analyzed in detail by using the statistical analysis tool SPSS.

A. Subjects

The study concerns 4 EFSM models which come from previous model-based studies [1], [20]. Each model has two versions. One includes EXIT state and other is free of EXIT state, denoted by the corresponding model name following $\_noexit$. Table I presents summary information concerning the subjects, including the model's size in terms of the number of states, the number of transitions and a brief description.

Table I
Experimental Models

Models	Number of States	Number of Transitions	EXIT State	Description
ATM	9	23	Yes	Automated
ATM *_noexit*	9	24	No	Teller Machine
Cashier	12	21	Yes	Cashier
Cashier*_noexit*	12	22	No	Machine
CruiseControl	5	17	Yes	Cruise Control
CruiseControl*_noexit*	5	18	No	System
FuelPump	13	25	Yes	Fuel Pump
FuelPump*_noexit*	13	26	No	System

B. Test generation system

In order to achieve the automatic test data generation and evaluate the efficiency for feasible transition paths in EFSM models, we develop a test data generation system for EFSM models using GA. The system supports not only the test generation of integer and real data types, but also non-numerical types such as Boolean and characters types.

Figure 2 shows the test data generation algorithm using GA. The implementation consists of two steps.

TestGeneration$(efsm, popsize, Imax, Pc, Pm)$:
Input : $efsm$: EFSM model to be tested
$popsize$: Population size
$Imax$: Maximum iteration number
Pc: Crossover probability
Pm: Mutation probability
Output: a set of test data for $efsm$
Create feasible transition paths
Randomly generate initial population $popu$
repeat $fitness \leftarrow evaluate(popu)$
$popu \leftarrow select(popu, fitness)$
$popu \leftarrow crossover(popu, fitness, Pc)$
$popu \leftarrow mutate(popu, fitness, Pm)$
$fitness \leftarrow evaluate(popu)$
$popu \leftarrow survive(popu, fitness)$
until $success$ or $iterationnumber > Imax$

Figure 2. Test data generation algorithm

Step 1: generate complete transition paths

First, complete paths with a variety of lengths are created according to breadth-first search technique, and potential feasible paths are produced by deleting the paths where condition conflicts exist. The path varies in length from 3 to 50. For each potential feasible path, a sequence of events that triggers all transitions in order is extracted and the types of input parameters of the events are identified.

Step 2: test data generation using GA

In this step, for each potential feasible path, find the input parameter values that trigger the path by applying GA. The initial population is generated randomly depending on their data types. The chromosomes are real-encoded, each gene representing one input parameter. Each individual is evaluated by a fitness function. A recent survey on search-based test data generation [24] suggests the notion of *approach level* and *branch distance* in order to construct a fitness function. The approach level will evaluate how close a chromosome is to the given path. The branch distance will measure how close is the first unsatisfied pre-condition to being true. So, in our research ,the following fitness function is applied.

$$fitness = approach\_level + norm(d)$$
$$norm(d) = 1 - 1.001^{d}$$

Where d is a branch distance, and $norm(d)$ is the branch distance value scaled between [0, 1].

In selection procedure $select(population, fitness)$, the parents are chosen according to their fitness values. This guarantees that the chromosomes with a higher fitness value have a higher likelihood of being selected. In crossover procedure $crossover(population, fitness, Pc)$, we produce

new offsprings selected by crossover rate Pc based on following computing inspired from [22].

$$y_1 = |0.05 \times (x_1 - x_2) + x_1|$$
$$y_2 = |0.05 \times (x_2 - x_1) + x_2|$$

Where x_1 and x_2 are the chosen parent individuals, y_1, y_2 are new individuals after applying the crossover operation, and a fixed 0.05 is chosen since it supplies a better score than other values randomly selected. In mutation procedure $mutate(population, fitness, Pm)$, an individual is chosen by mutation rate Pm, and a new gene will be generated randomly according its data type to substitute the original. After these procedures, population will be evaluated again, and a basic survive procedure $survive(popu, fitness)$ is employed to pick up certain individuals of the offspring into the next generation according to their fitness which means that better individuals (higher fitness) have a better chance of being chosen.

In the test data generation for EFSM models using GA, the below arguments are set as follows:

Crossover probability $Pc = 0.7$
Mutation probability $Pm = 0.08$
Survival probability $Ps = 0.8$
Population Size $popuSize = 20$
Maximum iteration number $Imax = 5000000$

GA-based test data generation is an heuristic process. When a new input is created, the EFSM model under the test has to be executed again in order to evaluate its fitness value. The cost of test generation algorithm depends mainly on the number of times the fitness function must be evaluated, i.e., the number of times the EFSM model is executed. In addition, the empirical experiments were done on two PCs and a SUN SPARC station, the time of test data generation varied with the different computers. Therefore, only the number of evaluation (NE) of fitness function during generating a test case for a FTP is considered as the cost of test generation. Thus, in this paper, the efficiency of the test data generation system is examined by the number of evaluations of the fitness function.

C. Metrics

In order to investigate which factors affect the performance of test data generation in EFSM, the following metrics are considered in this paper.

1) Length of path (LP): The number of transitions in a path.
2) Number of variables (NV): The number of variables defined or used in a path, including variables used as input parameters (defined) in events, defined in actions, used in conditions or actions on the path.
3) Number of variables defined in event (NVDE): The number of variables appeared as input parameters in event sequence within a path. These variables also are called event variables.
4) Number of variables defined in actions (NVDA): The number of variables defined in actions within a path.
5) Number of variables used in conditions (NVUC): The number of variables used in conditions within a path.
6) Number of variables used in actions (NVUA): The number of variables used in actions within a path.
7) Number of variables defined in event and used in conditions (NVDEUC): The number of variables defined in events of transition t_i, used in conditions of transition t_i or t_j within a path, and there are no other definitions with respect to the variables from transition t_i to t_j if used in t_j.
8) Number of variables defined in actions and used in conditions (NVDAUC) : The number of variables defined in actions of transition t_i, used in conditions of transition t_j within a path, and there are no other definitions with respect to the variables from transition t_i to t_j.
9) Number of conditions (NC): The number of nonempty conditions in a path.
10) Number of sub-conditions (NSC): The number of sub-conditions in a path.
11) Number of equal operators in conditions (NEOC): The number of equal operators in conditions within a path, including logical equal and numerical equal. Logical equal implies equalling to $True$ or $False$, and numerical equal means equal in integer, real or character value, not in logical value.
12) Number of numerical equal operator in conditions (NNEOC): The number of numerical equal operators in conditions within a path.
13) Length of path with event variables (LPEV): The number of transitions whose event is provided with nonempty input parameters within a path.
14) Number of numerical variables (NNV): The number of numerical variables defined or used in a path, including variables defined in events or actions, used in conditions or actions on the path. Numerical variables imply that their data type is integer or real.
15) Number of numerical event variables (NNEV): The number of numerical event variables within a path.

D. Data Collection and Description

In order to investigate and illustrate the effectiveness of the test data generation system for EFSM models, we have conducted a substantial number of experiments for the 8 EFSM subjects presented in Table I. For each subject, test cases are generated for potential (K) complete FTPs with different lengths. If test generation fail on a path within maximum iteration number, another path with the same length is chosen. In addition, corresponding numbers of evaluation of the fitness function as well as above metrics

are recorded during the test generation. Considering the randomicity of initial population in GA, we delete the highest and lowest number of the evaluation in the test cases for a path. Table II provides the summary statistics of 7 factors from the above metrics due to the limitation of page space.

IV. Experimental Results and Discussion

In this section, we examine the association between the number of evaluations and above metrics by using correlation analysis to identify which factors affect test generation efficiency for EFSM models. Then, we inspect the potential relationship between test generation efficiency and the factors by applying linear or nonlinear regression analysis. Finally, the empirical results are discussed.

A. Key factors identification

As displayed in Table II, the largest set is provided with 1917 data, and the smallest is 619 for all 8 EFSM models. The number of data points in each dataset is limited, and Spearman's correlation coefficient that is more robust to atypical values and to non-linearity of the underlying relationship, is used in this study.

In Section III-C, we introduced 15 metrics that are potential factors on affecting performance of test data generation. However, correlation may exist between these factors. For example, a longer path that contains more transitions may have more variables in the events and more conditions. Therefore, we first investigate the correlation between these metrics. Table III and Table IV list the correlation coefficients quantifying the strength of the interaction among the metrics in ATM and CruiseControl model, respectively. All of the values, except ones with + (meaning there is no significant correlation at 0.01 and 0.05 significant level) and - (denoting the correlation coefficients cannot be computed because NNEOC is constant) in Table III, are significant at $\alpha = 0.01$ level (2-tailed). The **bold-face** values representing the correlation coefficient are very high (larger than 0.9), implying there is a strong relationship between factors. In Table IV, most of metrics in CruiseControl model are significantly correlated with each other indicated by many **bold-face** values. However, this is not shown in Table III of ATM model, where there are few **bold-face** values.

To investigate the relationship between the number of evaluations (NE) during test generation and these metrics, the correlation coefficients are computed and shown in the last line in Table III and Table IV. It can be seen that there are strong correlations between NE and various metrics except NVDA, NVDAUC, NEOC and NNEOC (about 0.668) in CruiseControl. Again this is not shown in ATM where the coefficient is about 0.363, because of the low correlation between the metrics.

Matthew et al. [14] suggested that the length of path affects the performance of test generation for Ada software system. It is interesting to investigate whether does it exist in EFSMs. Figure 3 and Figure 4 display the relationships between NE and LP on 4 models with EXIT state and 4 models without EXIT state, respectively. It is observed, from Figure 3.(d) and Figure 4.(d), that there are no distinct relationships between NE and LP for FuelPump*_exit/noexit* models. However, Figure 3.(c) and Figure 4.(c) indicate that NE increases approximately exponentially with LP for CruiseControl*_exit/noexit* models. It can also be observed that NE increases in fluctuation as LP enlarges for other models. To obtain a more detailed insight, we analyze carefully the experimental results, and find that the number of numerical equal operators in the conditions (NNEOC) of FuelPump*_exit/noexit*, ATM*_exit*, Cashier*_exit*, ATM*_noexit*, Cashier*_noexit*, CruiseControl*_noexit* and CruiseControl*_exit* model, is 0, 0, 1, 1, 2, 2, 3 and 4, respectively. The corresponding correlation coefficients between NE and LP are -0.013+, 0.030+, 0.404, 0.323, 0.580, 0.676, 0.890 and 0.951, respectively. It suggests that the correlation between the NE and LP rises along with the growing of NNEOC.

On the other hand, when NNEOC is considerable, such as CruiseControl*_exit/noexit* model, there is a close relationship between NE and LP, and most of the other metrics strongly associated with LP (see the second column of Table IV). It seems to indicate that NNEOC has an important effect on EV. However, this is not explored by the analysis of the correlation between NE and NNEOC. Further inspection of the data reveals that NNEOC may be too small and insignificant to demonstrate the relationship. As shown in Figure 5, NE increases along with the growth of NNEOC on all models whose NNEOC is larger than 1. In addition, we especially create some complete paths on ATM*_noexit* model to improve NNEOC by paying special attention to transition T4 where a numerical equal operator is required. The largest NNEOC is equal 8 in the paths, and the correlation coefficients between NE and all metrics are displayed in Table V. It is obvious that NNEOC is significantly correlated with EV, holding the coefficient 0.887 close to the largest 0.899.

Table V
Correlation on T4 in ATM

Metric	LP	NV	NVDE	NVDA	NVUC
NE	**0.841**	**0.890**	**0.899**	0.311	0.755
Metric	NVUA	NVDEUC	NVDAUC	NC	NSC
NE	0.311	**0.838**	0.276	**0.821**	0.746
Metric	NEOC	NNEOC	LPEV	NNV	NNEV
NE	**0.840**	**0.887**	**0.880**	**0.890**	**0.881**

How about the number of equal operators in the conditions? Does it affects the efficiency of test generation for EFSM models? To address these questions, we analyze the NEOC on all EFSM models, and find there is little connection between NE and NEOC. This is easy to understand

Table II
DESCRIPTIVE STATISTICS OF DATA

Models	N	LP		LPEV		NV		NNEV		NEOC		NNEOC		NE	
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
ATM	833	4	30	2	9	5	13	4	11	1	12	1	1	8	2505
Cashier	1007	4	30	1	15	2	20	1	17	1	9	1	1	115	175281
CruiseControl	663	3	20	1	18	6	30	3	25	2	8	0	3	2	542810
FuelPump	1917	11	50	2	2	8	8	5	5	2	2	0	0	1	1446
ATM _*noexit*	1008	4	30	2	9	5	15	4	13	1	11	1	2	12	70901
Cashier_*noexit*	960	5	30	1	13	2	20	1	17	1	7	1	2	6	8805
CruiseControl_*noexit*	619	3	20	1	15	6	30	3	25	2	11	0	2	1	43907
FuelPump_*noexit*	1773	12	50	2	4	8	13	5	10	2	3	0	0	1	1470

Table III
CORRELATION ANALYSIS OF ATM MODEL

Metrics	LP	NV	NVDE	NVDA	NVUC	NVUA	NVD EUC	NVD AUC	NC	NSC	NEOC	NN EOC	LPEV	NNV	NNEV
LP	1														
NV	0.614	1													
NVDE	0.616	**0.999**	1												
NVDA	0.787	0.702	0.700	1											
NVUC	0.327	0.835	0.831	0.466	1										
NVUA	0.838	0.876	0.878	0.835	0.573	1									
NVDEUC	0.056+	0.667	0.660	0.190	0.880	0.315	1								
NVDAUC	0.703	0.658	0.635	0.786	0.633	0.738	0.280	1							
NC	0.825	0.771	0.769	0.783	0.568	0.888	0.332	0.724	1						
NSC	0.765	0.825	0.822	0.740	0.670	0.871	0.465	0.691	**0.986**	1					
NEOC	0.861	0.682	0.683	0.795	0.429	0.877	0.144	0.721	**0.979**	**0.934**	1				
NNEOC	-	-	-	-	-	-	-	-	-	-	-	-			
LPEV	0.616	**0.999**	1	0.700	0.831	0.878	0.660	0.653	0.769	0.822	0.683	-	1		
NNV	0.614	1	**0.999**	0.702	0.835	0.876	0.667	0.658	0.771	0.825	0.682	-	**0.999**	1	
NNEV	0.616	1	1	0.700	0.831	0.878	0.660	0.653	0.769	0.822	0.683	-	1	**0.999**	1
NE	0.404	0.363	0.365	0.364	0.242	0.405	0.124	0.339	0.393	0.384	0.390	-	0.365	0.363	0.365

Correlation is significant at the 0.01 level. +Correlation is not significant at the 0.01 and 0.05 level. -Cannot be computed because NNEOC is constant.

Table IV
CORRELATION ANALYSIS OF CRUISECONTROL MODEL

Metrics	LP	NV	NVDE	NVDA	NVUC	NVUA	NVD EUC	NVD AUC	NC	NSC	NEOC	NN EOC	LPEV	NNV	NNEV
LP	1														
NV	**0.995**	1													
NVDE	**0.994**	**0.998**	1												
NVDA	0.825	0.836	0.814	1											
NVUC	**0.993**	**0.992**	**0.990**	0.834	1										
NVUA	**0.994**	**0.988**	**0.984**	0.841	**0.989**	1									
NVDEUC	**0.995**	**0.992**	**0.991**	0.828	**0.999**	**0.989**	1								
NVDAUC	0.676	0.692	0.666	0.726	0.705	0.688	0.679	1							
NC	**0.995**	**0.988**	**0.986**	0.832	**0.992**	**0.997**	**0.993**	0.686	1						
NSC	**0.990**	**0.989**	**0.987**	0.830	**0.997**	**0.983**	**0.997**	0.693	**0.987**	1					
NEOC	0.641	0.643	0.658	0.383	0.660	0.600	0.665	0.465	0.615	0.669	1				
NNEOC	0.689	0.670	0.679	0.433	0.681	0.675	0.684	0.560	0.686	0.674	0.884	1			
LPEV	1	**0.995**	**0.994**	0.825	**0.993**	**0.994**	**0.995**	0.676	**0.995**	**0.990**	0.641	0.689	1		
NNV	**0.992**	**0.986**	**0.980**	0.852	**0.990**	**0.997**	**0.990**	0.699	**0.998**	**0.985**	0.592	0.667	**0.992**	1	
NNEV	**0.995**	**0.988**	**0.986**	0.832	**0.992**	**0.997**	**0.993**	0.686	1	**0.987**	0.615	0.686	**0.995**	**0.998**	1
NE	**0.951**	**0.943**	**0.943**	0.781	**0.946**	**0.945**	**0.947**	0.668	**0.949**	**0.943**	0.644	0.697	**0.951**	**0.945**	**0.949**

Correlation is significant at the 0.01 level

since logical equal operators are included in NEOC, which requires little effort to meet. Consequently, a conclusion can be drawn from the above analyses that NNEOC plays a key effect on the number of evaluations during test generation. That is to say, NNEOC of a path affects the performance of test data generation decisively.

B. Regression models analysis

In Section IV-A we concluded that the number of the evaluations has a close correlation with the number of numerical equal operators in the conditions on feasible

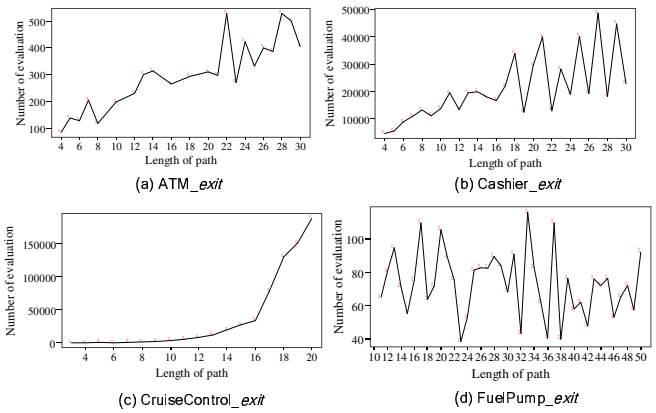

Figure 3. Relationship between NE and LP for models_*exit*

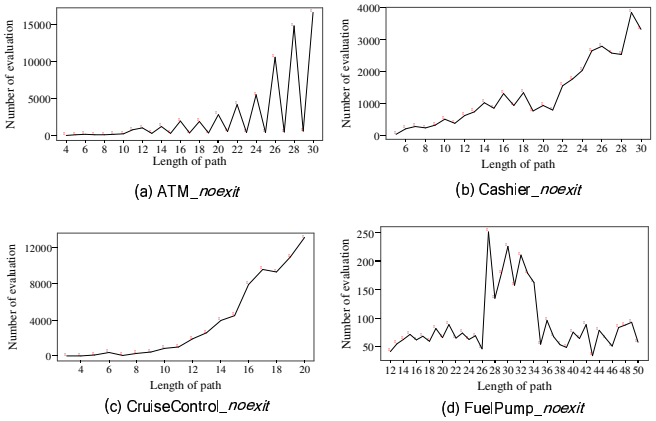

Figure 4. Relationship between NE and LP for models_*noexit*

transition paths.

However, what is this relationship? Is there a constant upward or downward trend that follows a straight-line pattern or a curved pattern? How about the metrics, length of path and number of variables in events? Are there similar relationships with the number of evaluation in EFSM models as in Ada software systems? To address these questions, we apply a regression analysis to the data of all the 8 subjects. The regression analysis on the correlation between EV and NNEOC is illustrated in Figure 6 on all models that NNEOC is larger than 1. It can be seen that the exponential coefficient of determination R-Square improves from 0.330 to 0.751 when NNEOC increases from 2 to 8. That is to say, the exponential regression model works better when NNEOC is considerable.

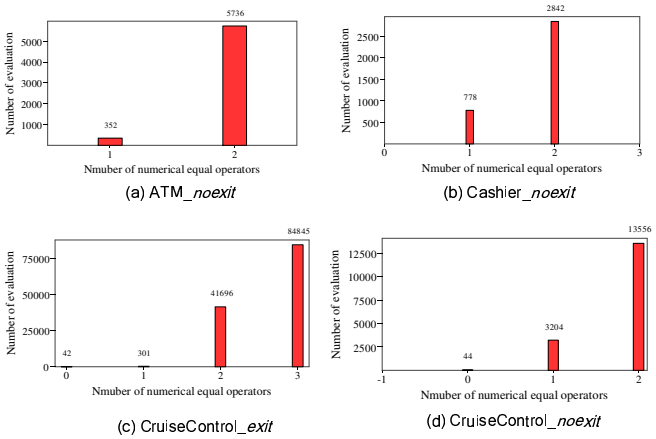

Figure 5. Relationship between NE and NNEOC for models with NNEOC>1

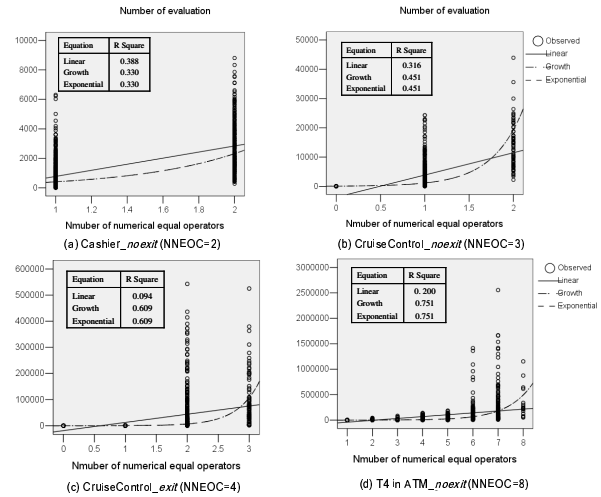

Figure 6. The regression analysis on NE and NNEOC on models with NNEOC>1

Considering a test data required to traverse a FTP, the number of transitions with event variables could be more reasonable than the number of transitions in the efficiency analysis of test generation. For example, the comparison LP with LPEV with respect to NE is given in Figure 7(a) for ATM model. It is evident that LPEV is more advisable. On the other hand, it is easy to understand that the number of numerical event variables is more reasonable than the number of variables defined in events since NVDE includes logical variables which are effortless to meet the test requirements. The advantage is clear in Figure 7(b), which displays the comparison NVDE with NNEV with respect to NE for CruiseControl_*noexit* model. So, we use LPEV and NNEV, instead of LP and NEV in following analysis.

In order to answer the question what relationship exists between EV and LPEV as well as NNEV, we apply regression analysis to all the models. The results are shown in Figure 8, Figure 9 and Figure 10, respectively, according to their NNEOC values. The scatterplot graphs exhibit a gradually strong exponential relationship between EV and LPEV (see the left hand side of the graphs) as well as NNEV (see the right hand side) when NNEOC varies from 0 to 4 depending

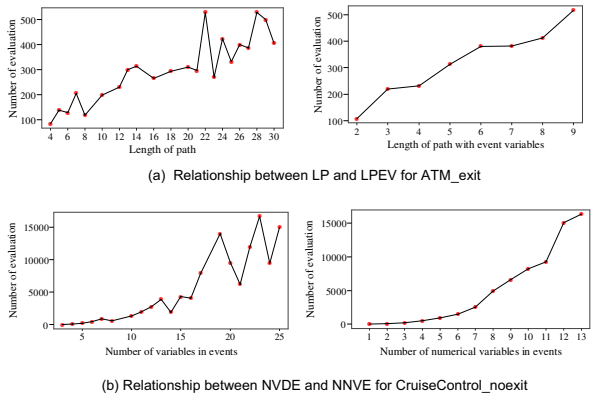

Figure 7. LP *VS* LPEV and NVDE *VS* NNVE

on models. The detailed coefficients of determination R-Square about exponential, increase from 0.053 to 0.855, and from 0.049 to 0.851 with respect to LPEV and NNEV, respectively. At the same time, we compute their R-Square by linear regression on corresponding models. It is further observed that the R-Square of linear is obviously smaller than that of the exponential on each corresponding model except FuelPump_*noexit* models, which NNEOC=0, shown in Figure 8(a). Especially, for the CruiseControl model (see the Figure 10(b)), their R-Square of exponential are 0.855 and 0.851, whereas R-Square of linear are 0.382 and 0.383, with respect to LPEV and NNEV, respectively. That is to say, the exponential regression models work well. As a result, we can draw the conclusion that there exists strong exponential relationships between EV and LPEV as well as NNEV when NNEOC is sufficient.

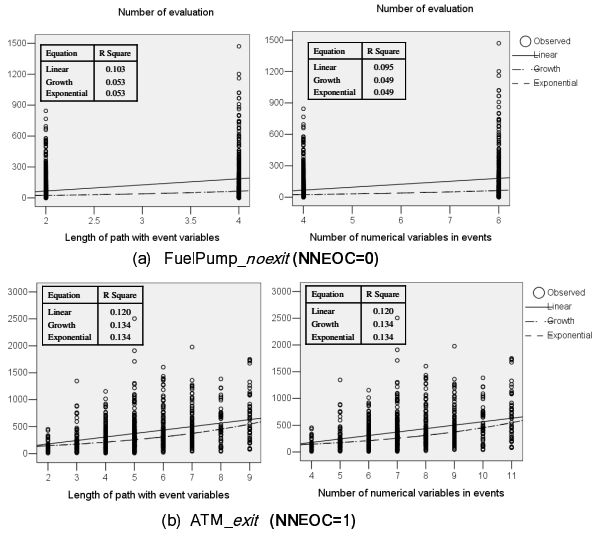

Figure 8. Regression analysis on models with NNEOC=0,1

It is distinctly different from Matthew's finding that there

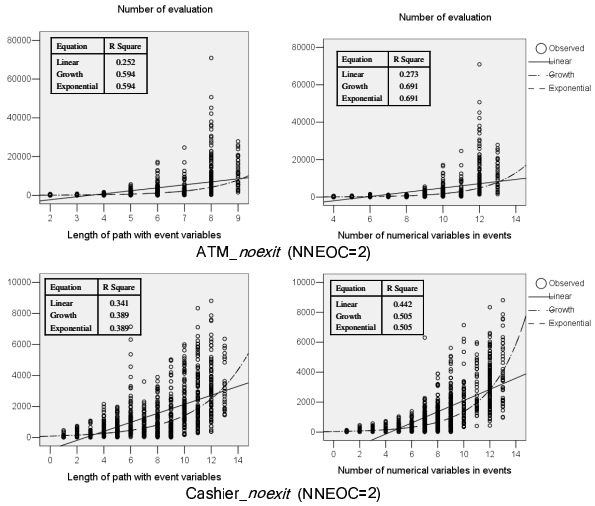

Figure 9. Regression analysis on models with NNEOC=2

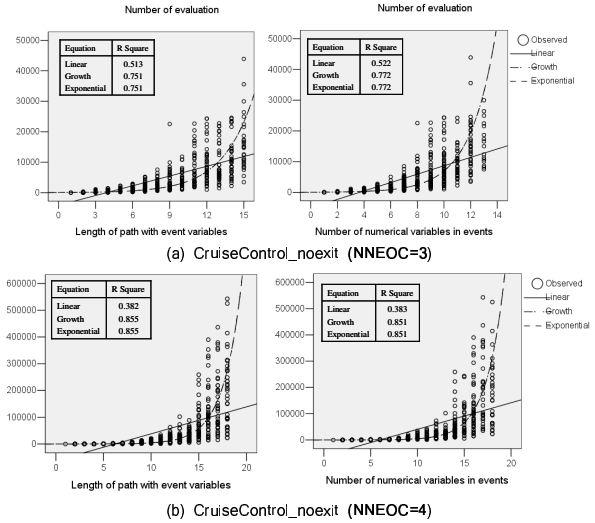

Figure 10. Regression analysis on models with NNEOC=3,4

was an approximate linear relationship between NE with NV generated in Ada software systems. In order to further support our proposition that NNEV has an exponential effect on NE, we plot the relationship between NNEV and LPEV. As demonstrated in Fig 11, there is a very strong linear relationship between NNEV and LPEV on all models. Their coefficient of determination R-Square are about 0.929 (92.9%), which is substantial, except Cashier_*noexit* model (about 0.792). Therefore, NNEV and LPEV should maintain similar relationship with EV. That is, there exists an exponential connection between NNEV and NE.

V. Related Work

Many test generation approaches had been studied in the literature for systems modelled as EFSMs [4], [5], [7], [8],

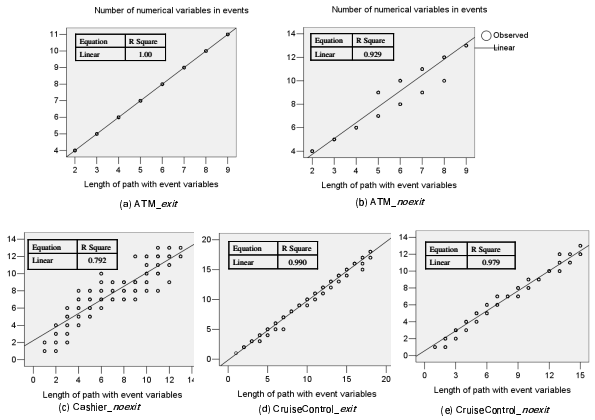

Figure 11. The Linear relationship between NNEV and LPEV for all models

[10], [18], [23]. Most of them focus on the generation of test sequences. Juhan et al. [10] presented a way to generate test sequences from EFSM models using a guided model checker. Duale et al. [8] introduced a method that overcomes the feasibility problem in advance. The method converted a class of EFSMs into consistent EFSMs in which all transition paths were feasible. Derderian et al. [5] proposed a GA approach to generate feasible paths from EFSM model. The approach evaluates the feasibility of a given transition path according to the number and the types of guards found in that TP. Kalaji et al. [18] reported a fitness metric to estimate the path's feasibility and to direct the search towards paths that are relatively easy to trigger. However, the studies do not tackle the problem of generating input test data to be used in testing the generated paths.

A number of comparisons and evaluations of testing efficiency have been carried out by researchers [2], [11], [12], [17], [25]. Ntafos [25] compared branch coverage, random testing and required pair coverage with 14 FORTRAN sub-routines. Frankl et al. [11], [12] conducted a study which compared the all-edges criterion to the all-uses criterion, and branch coverage and all-uses coverage criteria for nine Pascal programs. Hutchins et al. [17] reported an experimental study investigating the effectiveness of two code-based test adequacy criteria for identifying sets of test cases that detect faults. The all-edges and all-DUs coverage criteria were applied to 130 faulty program versions derived from seven moderate size base programs by seeding realistic faults. Another study evaluated a complete set of test cases to cover all possible paths in the code, for example mutation test techniques [26] or genetic test algorithms [28]. Above researches focus on comparing and evaluating the effectiveness and efficiency of different coverage criteria on various source code levels. It is clear that there is no similarity to our empirical study on test generation and efficiency for EFSM models.

VI. Conclusion and Future Work

Testing from formal specifications offers a simpler and more rigorous approach to developing test data. However, current work in specification-based automatic testing has been limited largely to FSM models in which context variables do not exist. It is hard to find a set of test data to trigger a given feasible path in EFSM models if more variables are used in the path and conditions are complex. Therefore, in this paper, we have presented a GA-based system to automatically generate test data for feasible transition paths in EFSM models.

In order to investigate the effectiveness of our test generation approach and identify the key factors affecting the efficiency of test generation in EFSM models, an empirical study has been conducted on 8 EFSM models, and the results were analyzed in detail using statistical analysis. We conclude that NNEOC plays a very important role in test generation efficiency. It is only when NNEOC is considerable that NE grows exponentially along with the increasing of NNEV or LPEV. Moreover, there is a very strong linear relationship between NNEV and LPEV. The results provide a significant guide to predict the testing efficiency based on NNVE or LPEV on a FTP for EFSM models.

Future research work on this topic will concentrate on the problem of test data generation in EFSM models including string, compound data types as well as function calls etc. The questions such as what relationship exists between the test generation cost and length of string, the number of string variables, as well as the number of function calls will be explored carefully.

Acknowledgement

The work described in this paper was supported by Beijing Natural Science Foundation under Grant No.4072021 and National Natural Science Foundation of China under Grant No.60903002. The authors wish to thank Lorna Anderson for help with proof reading.

References

[1] K. Androutsopoulos, N. Gold, M. Harman, Z. Li, and L. Tratt. A theoretical and empirical study of efsm dependence. In *IEEE International Conference on Software Maintenance, 2009. ICSM 2009.*, pages 287–296, Sept. 2009.

[2] V. R. Basili and R. W. Selby. Comparing the effectiveness of software testing strategies. *IEEE Trans. Softw. Eng.*, 13(12):1278–1296, 1987.

[3] L. C. Briand, Y. Labiche, and Y. Wang. Using simulation to empirically investigate test coverage criteria based on statechart. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 86–95, Washington, DC, USA, 2004. IEEE Computer Society.

[4] K.-T. Cheng and A. S. Krishnakumar. Automatic generation of functional vectors using the extended finite state machine model. *ACM Trans. Des. Autom. Electron. Syst.*, 1(1):57–79, 1996.

[5] K. Derderian, R. M. Hierons, M. Harman, and Q. Guo. Generating feasible input sequences for extended finite state machines (efsms) using genetic algorithms. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1081–1082, New York, NY, USA, 2005. ACM.

[6] K. Derderian, R. M. Hierons, M. Harman, and Q. Guo. Automated unique input output sequence generation for conformance testing of fsms. *Comput. J.*, 49(3):331–344, 2006.

[7] A. Y. Duale and M. U. Uyar. Generation of feasible test sequences for efsm models. In *TestCom '00: Proceedings of the IFIP TC6/WG6.1 13th International Conference on Testing Communicating Systems*, page 91, Deventer, The Netherlands, The Netherlands, 2000. Kluwer, B.V.

[8] A. Y. Duale and M. U. Uyar. A method enabling feasible conformance test sequence generation for efsm models. *IEEE Trans. Comput.*, 53(5):614–627, 2004.

[9] K. El-Fakih, N. Yevtushenko, and G. von Bochmann. Fsm-based incremental conformance testing methods. *IEEE Trans. Software Eng.*, 30(7):425–436, 2004.

[10] J. P. Ernits, A. Kull, K. Raiend, and J. Vain. Generating tests from efsm models using guided model checking and iterated search refinement. In *FATES/RV*, pages 85–99, 2006.

[11] P. G. Frankl and S. N. Weiss. An experimental comparison of the effectiveness of the all-uses and all-edges adequacy criteria. In *TAV4: Proceedings of the symposium on Testing, analysis, and verification*, pages 154–164, New York, NY, USA, 1991. ACM.

[12] P. G. Frankl and S. N. Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. *IEEE Trans. Softw. Eng.*, 19(8):774–787, 1993.

[13] S. Fujiwara, G. von Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Trans. Softw. Eng.*, 17(6):591–603, 1991.

[14] M. J. Gallagher and V. Narasimhan. Adtest: A test data generation suite for ada software systems. *IEEE Transactions on Software Engineering*, 23:473–484, 1997.

[15] A. Gupta and P. Jalote. An approach for experimentally evaluating effectiveness and efficiency of coverage criteria for software testing. *STTT*, 10(2):145–160, 2008.

[16] R. M. Hierons. Avoiding coincidental correctness in boundary value analysis. *ACM Trans. Softw. Eng. Methodol.*, 15(3):227–241, 2006.

[17] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *ICSE '94: Proceedings of the 16th international conference on Software engineering*, pages 191–200, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[18] A. S. Kalaji, R. M. Hierons, and S. Swift. Generating feasible transition paths for testing from an extended finite state machine (efsm). In *ICST '09: Proceedings of the 2009 International Conference on Software Testing Verification and Validation*, pages 230–239, Washington, DC, USA, 2009. IEEE Computer Society.

[19] K. Kapoor and J. P. Bowen. Experimental evaluation of the tolerance for control-flow test criteria: Research articles. *Softw. Test. Verif. Reliab.*, 14(3):167–187, 2004.

[20] B. Korel, G. Koutsogiannakis, and L. H. Tahat. Model-based test prioritization heuristic methods and their evaluation. In *A-MOST '07: Proceedings of the 3rd international workshop on Advances in model-based testing*, pages 34–43, New York, NY, USA, 2007. ACM.

[21] D. Lee and M. Yannakakis. Testing finite-state machines: State identification and verification. *IEEE Transactions on Computers*, 43:306–320, 1994.

[22] R. Lefticaru and F. Ipate. Automatic State-Based Test Generation Using Genetic Algorithm. In *Ninth International Symposium on Symbolic and Numeric Algorithm for Scientific Compuing*, pages 188–195, 2007.

[23] X. Li, T. Higashino, M. Higuchi, and K. Taniguchi. Automatic generation of extended uio sequences for communication protocols in an efsm model. In *IWPTS '94: 7th IFIP WG 6.1 international workshop on Protocol test systems*, pages 225–240, London, UK, 1995. Chapman & Hall, Ltd.

[24] P. McMinn. Search-based software test data generation: a survey: Research articles. *Softw. Test. Verif. Reliab.*, 14(2):105–156, 2004.

[25] S. C. Ntafos. An evaluation of required element testing strategies. In *ICSE '84: Proceedings of the 7th international conference on Software engineering*, pages 250–256, Piscataway, NJ, USA, 1984. IEEE Press.

[26] A. J. Offutt, G. Rothermel, and C. Zapf. An experimental evaluation of selective mutation. In *ICSE '93: Proceedings of the 15th international conference on Software Engineering*, pages 100–107, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.

[27] A. Petrenko and N. Yevtushenko. Testing from partial deterministic fsm specifications. *IEEE Trans. Comput.*, 54(9):1154–1165, 2005.

[28] J. Wegener, H. Sthamer, B. F. Jones, and D. E. Eyres. Testing real-time systems using genetic algorithms. *Software Quality Control*, 6(2):127–135, 1997.