A Research for Executable Path Automatic Generation Method Based on EFSM

Biao Wu[1*], Qi-Wei Ge[2]

[1] The School of Science, Zhejiang Sci-Tech University, China.
[2] Faculty of Education, Yamaguchi University, Japan.

* Corresponding author. Tel.: +86057186843329; email: biaowuzg@zstu.edu.cn
Manuscript submitted August 8, 2022; revised September 18, 2022; accepted November 1, 2022.
doi: 10.17706/jsw.18.115-30

Abstract: Extended finite state machine (EFSM) is currently one of the most widely used model in the field of software testing. EFSM model is an enhanced model based on finite state machine (FSM). Automated test data generation is still a challenging problem due to the complexity of EFSM which extends the input and output parameters, context variables as well as the predicate condition. These reasons lead to conflict of the context variable with the enable conditions in the transition path. In order to avoid infeasible path generation, this paper proposes a method based on modified breadth first search to generate feasible transition path (MBFS-FTP). To solve the problem of state explosion in path generation, this paper converts state diagram to transition diagram on EFSM model. In order to make the EFSM static model can be driven execution, this paper uses UML model and generate executable model, so that implements EFSM execute dynamically. When using breadth-first search algorithm (MBFS) on every target transition to generate an executable transition path, the conflict detection algorithm is utilized to the transition path for conflict decision, avoiding the occurrence of an infeasible transition path. Considering target transition has multiple feasible transition paths, this paper combines suggested penalized value of definition-predicate-use (def-p-use) pair and length of feasible transition path, and then develops the measurement method of feasible transition path, and obtained a set of feasible transition path containing all of transitions. Through the experiment on two actual EFSM model, verifying the effectiveness of MBFS-FTP for feasible transition automatic generation, the experimental results show that MBFS-FTP can reduce the feasible transition path length, and make transition paths are more easily to be triggered at the times of generating test cases lately, furthermore, it can improve the efficiency of generate feasible transition path and save a lot of time.

Keywords: EFSM model, feasible path, collision detection, feasible measures

1. Introduction

There are statistics show that in traditional software development projects, software testing work usually accounted for more than 40% of the total workload of software development, in total cost of software development, test cost to account for 30% to 50% [1], software testing in software development has become an indispensable important segment.

In the process of software testing, one of the key issues is generation of test cases, as a difficult point of software testing, which affects the actual effectiveness of software testing. Model-based tests can be automatically generated test cases through the software requirements specification, which save the test time and test cost, therefore, this method attaches great importance to the industry and become a new challenge in the field of software testing [2]. Finite State Machine (FSM) and Extended Finite State Machine

(EFSM) are two of the most widely used models. EFSM expands on the basis of input and output parameters, context variables, defined in the context variable and input parameters on the predicate condition and operation on the foundation of FSM [3], and it can express control flow and also data flow section, thus EFSM is more suitable to describe the application of complex system. So, this paper uses EFSM as tested model for solving the problem of generating test cases.

In the test case generation technology based on EFSM, mainly includes two aspects: one is the automatic generation of test sequence; the other one is the automatic generation of test data. This paper is focus on the former.

At present, there have been many test sequence generation methods based on FSM [4-6], but there are still many challenges based on EFSM. Due to the mutual influence between data flow and control flow, some existing predicate conditions in transition path of EFSM model are not met, which leading to the path is infeasible, therefore, feasible path in EFSM model is an undecidable problem [7]. In addition, although there have some test data generation methods based on model [8-10], but test data generation technology is still immature based on EFSM.

In view of the automatic generation test sequence EFSM-based model, the key factor lies in feasibility analysis of transition path. Several kinds of methods of automatic generation of test sequence meet test coverage criteria based on data flow respectively are proposed in References [11]-[13] proposed. In order to solve the problem of the feasibility of transition path, Huang [14] proposed a method of the transition enforceability analysis, using this method, the states regarded as node, the transitions regard as edge, and using breadth-first search algorithm for extending EFSM model into a TEA tree, and thus generating an executable transition sequence. Hierons [15] used a method of expand the EFSM model method to avoid infeasible path problem, but the disadvantage of the method is likely to cause the problem of states explosion. Some internal researchers also successively put forward its own methods of automatically generated executable transition sequence, such as Zhao Baohua [16] put forward an improved method in transformation perform analysis, this method adopts the only input and output sequence in the control flow part, adopts full Definition-Use Path in the data flow part, and then use the depth first search strategy to generate TEA tree, then obtains test sequence. Yang Rui [17-18] proposes a path feasibility measurement method based on data flow analysis to predict the feasibility of the test sequence, this method combines the technique of static analysis and dynamic analysis, as far as possible to avoid the infeasible path, generate a more feasible path subset. But it is possible to cause the problem of state explosion. This paper used transition graph to generate test sequence, which avoided the problem of state explosion effectively. Shu Ting [19] proposed a test sequence generation approach for EFSM-based protocols conformance test by using the transition feasibility estimation. They designed a fitness function to guide the test generation with a trade-off among path feasibility, coverage criterion and path length and developed an adaptive exploration algorithm to generate executable test sequences through expanding CPs. Wang Weiwei [20] investigated the relationship between EFSM test data generation efficiency and its influence factors, build a multi-gene genetic programming (MGGP) predictive model to forecast EFSM test data generation efficiency according to the feasible transition paths of EFSMs.

After analyze the relevant contents of EFSM model, aiming at some existing disadvantages of methods of test cases generated automatically based on EFSM model, in this paper, the test sequence generation method of EFSM model were studied. The major contributions of this paper are as follows:

1) To obtain test transition paths from EFSM model, this paper defined a rule for converting state diagram to the transition diagram. The method can directly generate transition paths avoid the problem of state explosion.

2) This paper modified Breadth-First Search Algorithm (MBFS) to automatically generate test sequence from the initial transition to every other transition based on the transition diagram.

3) To make EFSM model having the dynamical executed ability, this paper adopts the method of UML modeling, takes advantage of state model analyzer (SMC) tool to establish the executable model.

4) In the MBFS, it is must detect and analysis the feasibility of the test sequence, this paper designed Conflict Checker Method (CCM) and Feasible Measure Method (FMM) to identify the test sequence. The two methods are based on the data flow and control flow for quantitative analysis the feasibility of the test sequence.

According to the experiment on two actual EFSM model, we verified the effectiveness of MBFS-FTP for feasible transition automatic generation. And the experimental results showed that MBFS-FTP can reduce the feasible transition path length, so that make transition paths are more easily to be triggered at the times of generating test cases lately, even it can improve the efficiency of generate feasible transition path and save a lot of time.

Rest of the paper is organized as follows. Section 2 mentions the description of EFSM model and focus on the Model conversion. Section 3 describes how to obtain test sequence including establish the executable model and modify BFS. The design and implement of CCM and FMM are given in section 4. And section 5 is design and analysis of simulation experiments. Finally, section 6 presents the conclusions and some suggestions for future work.

2. EFSM Model Conversion

2.1. Related Description and Definition of EFSM Model

An EFSM M can be expressed a six-tuple: (S, s_0, V, I, O, T), where: S is a finite set of states; $s_0 \in S$ is an initial state; V is a set of context variables; I is a finite set of input; O is a finite set of output; T is a finite set of transitions. Each transition t of set T is a six-tuple $T = (s_i, s_j, i_t, o_t, g_t, a_t)$ where s_i is the source state of transition t, s_j is the target transition of transition t, $i_t \in I$ is input operator, $o_t \in O$ is the output operator, g_t is predicate condition of the current value of variable, a_t is a series of operating statements of output or assignment, g_t and a_t are called the guard and the action in some paper.

EFSM model expands input and output parameters, context variables and the predicate condition and operation defined in the context variable and input parameters on the foundation of FSM model. An ATM (Automated Teller Machine) [21] EFSM model is shown in Fig. 1.

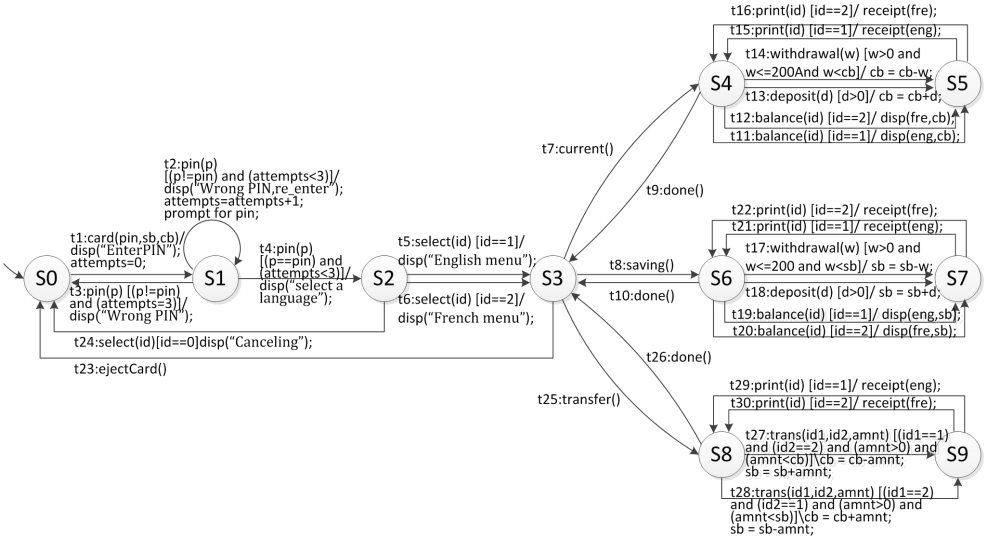

Fig. 1. EFSM model of ATM system [21]

Definition 1. Transition Path (TP): in EFSM model, for each target transition t_n, regardless of the predicate condition, if there is a transition sequence TP from the initial transition t_1 to target transition t_n, namely $TP = (t_1 \cdots t_n)$, which called TP is a transition path.

Definition 2. Transition Path Length (TPL): in EFSM model, a transition path length is defined as the number of transitions in a path, denoted as $TPL = |TP|$.

Definition 3. Test Sequence (TS): in EFSM model, the TS is the Feasible Transition Path (FTP), which is a transition path after the predicate condition to judge if the transition path is feasible, so the transition path is called a test sequence. For a transition path, if there are at least a set of test data driven EFSM execution when all the transitions of the transition path are covered orderly, said the transition path is the feasible transition path or test sequence.

Definition 4. Definition-Predicate-use (def-p-use): Existing def-p-use pair in a transition path $TP = (t_i \cdots t_j)$, if and only if the transition t_i in TP has assignment operations of variable v, the transition t_j in TP has the predicate used of variable v for judging, and the definition assignment of the variable v in the path from t_i to t_j does not exist again.

2.2. Model Conversion

General EFSM model diagram is shown in Fig. 1, every "circle" expresses the state of the software system, it does not contain any variables, the transitions between states includes variable operations and determine of predicate condition from one state to another state in software system. However, we need to get the transition paths in EFSM model at last instead of state paths.

To get transition path set of EFSM model, Yang, R. et al. [17] generates state path according to state path diagram of EFSM model, then consider transition between states, using a whole combination algorithm to convert state paths to transition path. Although this method is effective, but the complexity and time of the method for obtaining the transition path are increased. Therefore, this paper adopts the method of converting state diagram to transition diagram, directly obtained TPs, which will save a lot of time and manpower.

We defined transition directed graph and it can be obtained by transform state diagram, more convenient to using the improved first search algorithm for generating transition paths automatically later based on directed graph. Due to the complexity of the EFSM model diagram, consider the following when converting:

1) EFSM state model in general. Consider the following EFSM state model diagram with that the transition predicate condition, variable update, input and output are omitted, as shown in Fig. 2 (a):

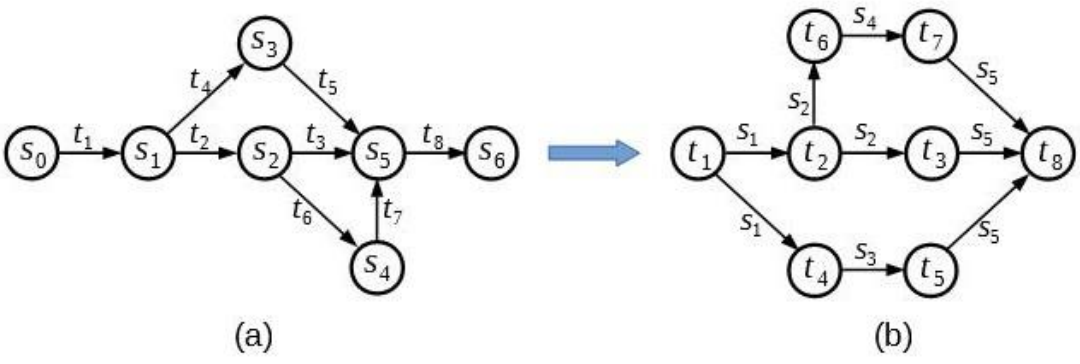

Fig. 2. General EFSM state diagram conversion

In this situation, during the conversion process, starting from the initial transition t_1 to the end of the transition t_8, each transitions is treated as a node, and each state is treated as the edge of each connection between transitions. The initial node represents the out-transition of the initial state of EFSM model, which

is the initial transition, terminal node is the terminal transition. The two nodes are adjacent to each other and are connected by an edge representing the state. Since the state is only a signal representing the location state of model and does not contain other useful information, so the converted EFSM transition diagram ignores the initial state s_0 and the ending state s_6, and the same as to other EFSM transition diagram. The converted EFSM transition diagram is shown in Fig. 2 (b).

2) EFSM state model with multi-transitions. In the actual EFSM, the two states may be converted by a variety of different conditions, resulting in the state may have a number of transitions to other state, as shown in Fig. 3 (a):

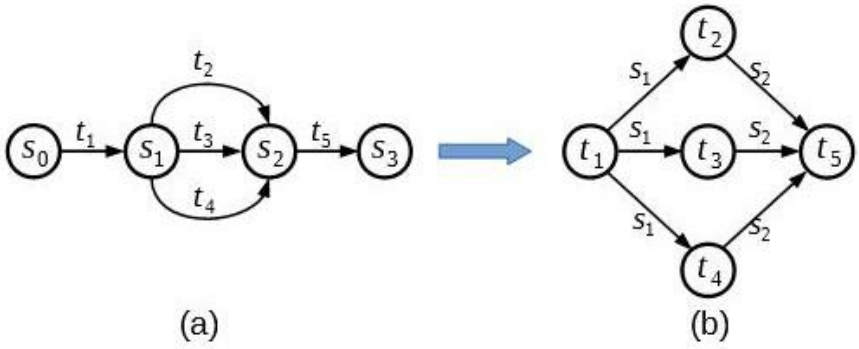

Fig. 3. EFSM state diagram conversion with multi-transitions

In the Fig. 3 (a), there are three transitions (t_2, t_3, t_4) between the states s_1 and s_2, where s_1 is the initial state, s_3 is the terminal state, t_1 is the initial transition, the converted EFSM transition diagram as shown in Fig. 3 (b), no matter how many transitions exist between s_1 and s_2, we only need to know that the path of the initial transition t_1 to reach these transitions by passes states.

3) EFSM state model with self-cycle. In the actual EFSM, we can also see the situation, when one EFSM in a certain state, after the “refresh” operation or "reset" operation, the EFSM is still at this state, so that the state exists the self-cycle. As shown in Fig. 4 (a):

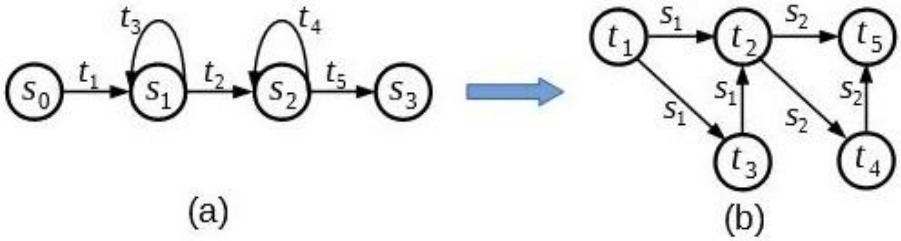

Fig. 4. EFSM state diagram conversion with self-cycle

In the Fig. 4 (a), the state s_1 and the state s_2 have self-cycle, where s_0 is the initial state, s_3 is the terminal state, t_1 is the initial transition, and the converted EFSM transition diagram is shown in Fig. 4 (b). The self-cycle transition of state s_1 is transition t_3, from the initial transition to t_3 through the state is s_1. Therefore, from the converted transition diagram, the path (t_1, t_3) through the state s_1, the path (t_3, t_2) is also through the state s_1.Therefore, when considering the case of self-cycle, it is only necessary to note that the state in which the self-cycle exists in the edge between the in-transition and the out-transition.

4) EFSM state model with loop. Most EFSMs exist the situation that one state returned to this state through other transitions, and therefore, there is a loop in the EFSM state diagram, as shown in Fig. 5 (a):

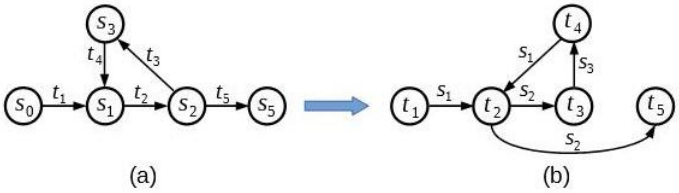

Fig. 5. EFSM state diagram conversion with loop

In the Fig. 5 (a), there is a loop between the states (s_1, s_2, s_3), no matter which three states can start and return to the state, where s_0 is the initial state and s_4 is the terminal state , t_1 is the initial transition. According to the principle of conversion that the transition as a node and the state as the edge, the converted transition diagram shown in Fig. 5 (b), we can see the converted transition diagram also have the loop between transitions correspondingly. But in the situation of generating transition paths, the loop will be considered only once.

3. Design of Feasible Paths Generation Method

3.1. Establish Executable EFSM Model

EFSM model is usually obtained according to the requirements of the system, in general, these models are static, and exists in the form of diagrams or abstract graphics, which does not possess dynamic behavior. In order to be able to obtain feasibility transition paths, and automatically generate test data subsequent, EFSM should be expressed in the form of an executable model. Therefore, this paper uses the executable model to represent the needed EFSM dynamic model. Executable model is a model system that can simulate the real system with dynamic behavior, executable models can be even used as a prototype system to test, and different from the static model, the executable model defines the dynamic behavior as part of the model. This paper using the State Machine Compiler (SMC) to establish executable EFSM model according to the form of UML modeling, converts it to a form of programming language description.

UML modeling of Executable EFSM model [17] as shown in Fig. 6:

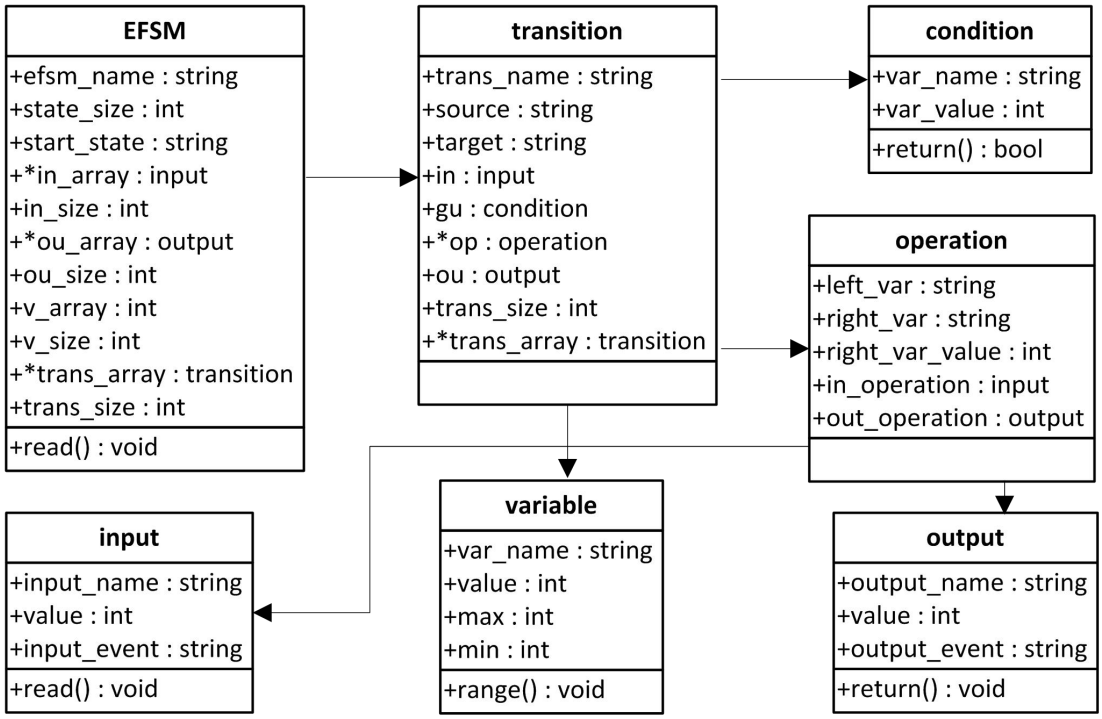

Fig. 6. EFSM class diagram [17]

3.2. Test Sequence Generation

Path Generation Description. In this paper, we improved the breadth-first search algorithm in directed graph, the method (MBFS-FTP) automatically generates the feasible transition path for EFSM model according to transition directed graph. Detailed steps as follows:

Step1: Analyzing state diagram of EFSM model, converting it to transition diagram, adopting the method of UML modeling, and establishing dynamic executable model.

Step2: For the initial value of input variables in each transition, using Modified Breadth First Search (MBFS) automatically generate feasible transition path.

Step3: Detecting the conflict of transition path and measuring the feasibilities of test sequences in step2.

Test Sequence Generation. EFSM transition diagram is a directed graph, how to obtain transition path based on directed graph, you will need to traverse the directed graph. Graph traversal belong to the contents of the data structure, every vertex from a graph, access to all other vertices only one at a time. Graph traversal is a basic operation, and many other methods of graph are based on its traversal operation. Graph traversal operation mainly has two basic methods, Depth-First Search Algorithm (DFS) and Breadth-First Search Algorithm (BFS).

This paper is to obtain transition path as short as possible from the initial transition to all other transitions. However, some paths may be triggered multiple times, which lead to each transition node may visit more than one times, so DFS is not suitable. Therefore, this paper designs MBFS to realize transition path automatically generation.

Because of each transition node of EFSM model may visited many times, so every node no longer needs to be marked tag, MBFS is shown in Algorithm 1:

Algorithm 1: MBFS

Input: initial transition t_{s0}, target transition t_{target}
Output: TS
BEGIN
Q[n]; //empty queue
Enqueue(Q,t_{s0});
WHILE count = false do
  Check(Q,t); //transition t
  FOR m<max iteration do
    FOR each next transition t_{next} adjacent to t do
      IF flag of CCM = true THEN //CCM: according to Algorithm 2
        Enqueue(Q,t_{next});
        Previous[$n_{t_{next}}$]=n_t; //location
        m++;
        IF t_{target} = t_{next} THEN
          count = true;
          break;
  END

In the algorithm 1, each transition will be as target transition in turn, seeking the executable shortest path from initial node to the destination node. For every target transition node, starting from the initial transition node, looking for adjacent nodes, and then use the conflict detection algorithm to determine whether the node and its adjacent conflict or not. If there is no conflict, then adding the node to the path queue, otherwise the neighboring node is given up, looking for the next adjacent node sequentially.

Eventually, the shortest executable path from the initial transition node to the target transition node stores in the form of queue.

4. Feasibility Analysis of Transition Path

4.1. Feasibility Decision of Transition Path

Due to context variables, operation and precondition decision in EFSM model, when the preconditions of a transition path containing these variables are not satisfied, the transition path will be infeasible. Therefore, in our method, once generating a transition path, we need decide the feasibility of generated transition path. In this paper, conflict checker algorithm is used to decide the feasibility of the transition path.

When using MBFS is to generate automatically transition path, we only considered the feasibility of the path in the directed graph regardless of predicate condition decision and variable update operations of transition in the actual EFSM model. Considering the EFSM model in the actual situation, in order to realize the infeasible transition path eliminated automatically, this paper designs a transition path feasibility decision algorithm, called Conflict Checker Method (CCM), as shown in Algorithm 2:

Algorithm 2: CCM

```
Input: current transition, previous transition
Output: flag
BEGIN
  Read input event(i_t); //include input variables
  v = random(); // variable value based on range
  flag = false;
  FOR each pair of variable and operator of current transition do
    IF left_var = v and left_var ∈ i_t THEN // left_var is the left variable of an assignment
      Set left_var according to operator;
      flag = true;
  return flag;
ELSE
  Check condition according to operator
  return flag;
END IF
  END FOR
  IF flag = true THEN
    Update variables and conditions;
    Update queue memory;
  END IF
END
```

4.2. Feasibility Measure of Transition Path

If there are multiple feasible transition paths for a target transition after the feasibility decision, thus a feasible measure is required to these feasible transition paths. In this paper, we mining the relationship between the predicate condition and the operation in these executable transition paths, take into account the length of the executable path, the type of operation and the type of guard, measure the feasibility of the transition path, and select higher executable transition path to generate test data for improving the

efficiency of test data generation. On the foundation of Kalaji [13] dependence analysis, we propose a method of Feasible Measure Method (FMM) based on the knowledge of data flow.

Previous studies shown that a shorter transition path is to be triggered with higher possibility in the case of the number of transitions is given, and it is easier to generate the required test data [22]. The longer transition path may have more predicate conditions and def-p-use pairs, resulting in the proposed penalty value may be larger, increasing the path length to balance the contradiction between the proposed penalty values. For each target transition, in order to find a path with shorter length and higher executable, we propose the following feasibility measure of transition path:

$$f = \begin{cases} |TP|^{\tau} \sum_{i=1}^{k} v(dp_i), & \sum_{i=1}^{k} v(dp_i) \neq 0 \\ |TP|, & \sum_{i=1}^{k} v(dp_i) = 0 \end{cases} \quad (1)$$

Where $|TP|$ is the length of transition path, $\tau(0 < \tau < 1)$ is the weight value of the influence of the path length to the probability of the transition path, used to balance the path length and the recommended penalty value, dp_i is the def-p-use pairs in transition path, $v(dp_i)$ represents the recommended penalty value for the DP-use pair between two transitions in the transition path, and k is the total number of DP-use pairs in transition path. Equation (1) shows that when the value of f is smaller, the feasibility of the test sequence is higher; on the contrary, the feasibility of the test sequence is lower. When the sum of the recommended penalty values for DP-use in a transition path is 0, that is $\sum_{i=1}^{k} v(dp_i) = 0$, indicating that there is no def-p-use pair dependency between all transitions in the test sequence, and the test sequence is most feasible, the shorter test sequence is more likely to be triggered, the value of f is set to $|TP|$.

After the feasibility analysis of transition path, for each target transition, there is a shortest executable path with higher feasibility from the initial transition to the target transition. And finally obtain an executable transition path set, which used to generate a better test data set in the later detection EFSM system.

5. Design and Analysis of Simulation Experiments

This paper adopts two actual EFSM model to validation test. We verify the effectiveness of the test sequence generation method compared to the traditional method based on simply control flow or data flow at first. And then verify the effectiveness of the test data generation method, using scatter search algorithm compared with the other methods to generate test data, such as genetic algorithm.

5.1. Experimental Model

In this paper, software environment using Eclipse 4.5 experiment platform + JDK1.7, hardware environment using Windows 10 64 bit operating system (R) Core (TM) i5-10500 CPU @ 3.10GHz. In order to verify the effectiveness of the method, chose two more typical actual EFSM model to make a series of experiments, respectively is: (Automated Teller Machine, ATM) [21] (as shown in Fig. 1), (Initiator Responder Protocol, INRES) [14] (as shown in Fig. 7).

The ATM model contains definite termination status, and INRES model has loops. In this experiment, the ATM model experiment is not set the end state of path must be the model termination status, because in EFSM model transition path, the initial state reaches the termination state is just a special case of the all transition paths.

Table 1. Experimental Target Model Properties

Target model	States	Transitions	Input parameters
ATM	10	30	6
INRES	4	16	2

Table 1 lists the general properties of two targets model, where “input parameters” column is the number of states, and “transition number” column is the target model contains the total number of transition, and “input parameters” column is the number of input parameters required in target model.

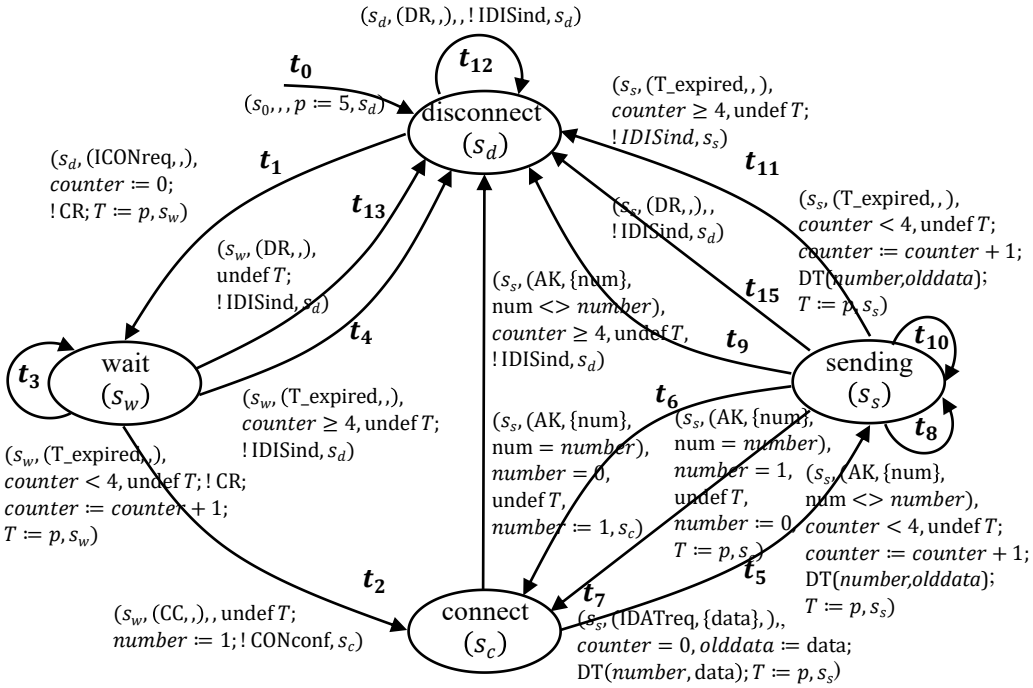

Fig. 7. INRES EFSM Model [14]

In order to verify the effectiveness of test sequence automatic generation method (MBFS-FTP), this experiment is aimed to answer the following questions in the research:

(Q1) How are validity and efficiency of MBFS-FTP?

(Q2) How is the effectiveness of CCM and FMM?

5.2. Effectiveness of Test Sequence Generation Method

1) To answer the validity of MBFS-FTP method in (Q1), we measure from two aspects:

A) Does the generated feasible path set all cover all transitions?

B) What is the length of the generated feasible path?

Since the EFSM model starts from the initial state until the end of the state. Therefore, this paper considers all transitions in the EFSM model as target transition, looking for a shorter set of feasible transition paths from the initial transition to the target transition, thus ensuring that the generated set of feasible paths covers all the transitions. The experimental results of using MBFS-FTP method, ATM target model and INRES target model to generate feasible path are shown in Table 2 and Table 3 (some feasible transition paths are listed in Table).

Table 2. ATM： Transition path for each transition

t_i	Path	Length	Time(ms)
t_1	t_1	1	31
t_2	t_1t_2	2	32
t_3	$t_1t_2t_2t_2t_3$	5	31
t_4	t_1t_4	2	31
…	…	…	…
t_{27}	$t_1t_4t_5t_{25}t_{27}$	5	47
t_{28}	$t_1t_4t_5t_{25}t_{28}$	5	31

t_{29}	$t_1t_4t_5t_{25}t_{27}t_{29}$	6	63
t_{30}	$t_1t_4t_5t_{25}t_{27}t_{30}$	6	47
Average		5.46	46.08

Can be seen from Table 2, ATM target model finally got 30 short test sequence and covers all the transitions in EFSM model. Due to the character of MBFS algorithm, each test sequence on the premise of feasible, guarantees has a minimum path length, each executable path there is no unnecessary transition, if there is a cycle, the cycle is executed only once according to the principle of circulating. Among them, the transition t_3 is the only transition appeared in the infeasible path, due to the operation (attempts = 0) of transition t_1 and the guard (attempts = 3) of transition t_3 conflict, therefore, the transition path (t_1t_3) is a infeasible path. Thus, in the path $(t_1t_2t_2t_2t_3)$ of table 2, the transition t_2 repeated three times to change update the value of the variable attempts to make attempts meet the guard condition of the transition t_3.

Table 3. INRES：Transition path for each transition

t_i	Path	Length	Time(ms)
t_0	t_0	1	29
t_1	t_0t_1	2	31
t_2	$t_0t_1t_2$	3	34
t_3	$t_0t_1t_3$	3	35
…	…	…	…
t_{12}	t_1t_{12}	2	29
t_{13}	$t_0t_1t_{13}$	3	31
t_{14}	$t_0t_1t_2t_{14}$	4	53
t_{15}	$t_0t_1t_2t_5t_{10}t_8t_{15}$	7	72
Average		4.74	41.25

Because of the INRES model represents a network request and response protocol, the circulation is more, so in order to ensure path length is small, circulating executed only once in accordance with the principles of. MBFS - FTP method is used to get the test sequence as shown in Table 3.

2) In order to answer the efficiency of MBFS-FTP method in (Q1), MBFS-FTP is compared with the method MOST proposed in [23], and the results are shown in Table 4:

Table 4. Experimental result of comparison between different approaches

Approach	ATM			INRES		
	Coverage (%)	Path length		Coverage (%)	Path length	
		max	averag		max	average
MBFS-FTP (ours)	100	8	5.46	100	7	4.74
MOST [23]	100	13	6.63	100	9	5.69
Reduction (%)	-	38.46	17.65	-	22.20	16.70

We can see from Table 4, in terms of the longest transition path, MBFS reduces 38.46% on the ATM model method and decreased by 22.2% on the INRES model compared with MOST, MBFS method has more advantages. In the full transition coverage rate, MOST and MBFS both have very good effect on the two target models, reached 100%. But in terms of average transition path length, MBFS reduces 17.65% on the ATM model and decreased by 16.7% on the INRES model compared with MOST. Therefore, in the case of transition coverage rate has reached 100%, the length of transition path generated MBFS is smaller, so in the case of the number of test sequence is certain. In the generation of test data, transition path generated by MBFS-FTP are more likely to be triggered.

3) In order to answer the effectiveness of CCM and FMM in (Q2), we measure from two aspects:

a) Using MBFS-FTP to generate feasible transition path, in the case of WITH and WITHOUT the CCM. Verify the impact of CCM on feasible transition paths.

b) Using MBFS-FTP with CCM to generate feasible transition path, in the case of USE and UNUSE the FMM and compared in terms of path length and feasible value, and the validity of the FMM will be tested at generating test data in the late based on feasible transition paths.

① In the case of a), drive two target EFSM models to be executed. In the case of WITH and WITHOUT the CCM, generate feasible transition paths. The comparison between the transition path length and the number of feasible transition paths is shown in Table 5 and Table 6:

Table 5. Comparison result of transition path generation between WITH and WITHOUT CCM (ATM)

Approach	ATM			
	Path length		No. of Path	
	max	average	feasible	infeasible
WITH	8	5.46	30	0
WITHOUT	9	6.15	17	13
Reduction (%)	11.11	11.22	43.33	

Table 6. Comparison result of transition path generation between WITH and WITHOUT CCM (INRES)

Approach	INRES			
	Path length		No. of Path	
	max	average	feasible	infeasible
WITH	7	4.74	16	0
WITHOUT	8	5.32	9	7
Reduction (%)	12.50	10.90	43.75	

As can be seen from Table 5 and Table 6, the use of conflict detection mechanism (WITH) has a greater degree of improvement than the use of the collision detection mechanism (WITHOUT), no matter where the path length or the number of feasible transition path. For example, consider the target model ATM, the maximum path length is 8 when with CCM, and the maximum path length is 9 when without CCM. So that it is easier to be triggered for feasible transition to generate test data in the late. In the case of average path length, WITH is only reduced 11.22% compared to WITHOUT in the ATM model and only reduced 10.90% in the INRES model. Therefore, the CCM has no obvious effect on reducing the path length. In the case of the number of feasible transition path, WITH has more 13 feasible paths in the ATM model than WITHOUT, and increased 43.33%, while in the INRES model the number respectively are 7 and 43.75%, the effect is very significant. Those means that the use of CCM can form more feasible transition path, and also makes the transition path is more likely to be triggered under generating test data

② In the case of b), using MBFS-FTP with CCM to generate feasible transition path, USE and NONUSE FMM compared the two aspects of the path length and the transition path feasibility value (τ takes 0.5 calculated according to the equation (1)), the compared results are shown in Table 7 and Table 8:

Table 7. Comparison result of transition path generation between USE and NONUSE FMM (ATM)

Approach	ATM			
	Path length		Path feasible value	
	max	average	max	average
USE	8	5.46	1229.84	278.29
NONUSE	9	5.83	1396.92	354.36
Reduction (%)	11.11	6.35	11.96	21.47

Table 8. Comparison result of transition path generation between USE and NONUSE FMM (INRES)

Approach	INRES			
	Path length		Path feasible value	
	max	average	max	average
USE	7	4.74	1035.34	225.32
NONUSE	8	5.09	1131.95	297.24
Reduction (%)	12.50	6.88	8.53	24.20

As can be seen from Table 7 and Table 8, the use of the feasibility measure method (USE) and the nonuse of feasibility measures (NONUSE) compared, the same result as ①, because of BFS algorithm characteristics in graph theory, the aspect of reducing the length of the transition path is no obvious effect, such as in the ATM model average path length USED only reduced by 6.35%, while in the INRES model, USE is only reduced by 6.88%, so in improving the length of the transition path, due to the inherent advantages of BFS, USE does not have any great effect, we can see the validity of MBFS selected in this paper.

In the comparison of the feasible value of transition path, because of the case of using CMM, the transition path generated is feasible. The FMM is used to determine the suitable feasible transition path when the target transition has multiple feasible paths, and finally select the transition path with low feasible value to generate test data in the late.

Since the type of guard in the transition of ATM model is g^{vc}, it may make the recommended penalty value of some transition paths large, resulting in a greater value of feasibility measure, which indicate the feasibility of transition paths are low. But there are a lot of "select (id)" such methods, making g^{vc} is easy to be met, so most of transition paths are still very high feasible, therefore, the feasibility measure value is relatively small, so it makes the average transition path feasibility value much smaller than the maximum value. The same situation also occurs in the INRES model, through such the expression "counter = counter + 1" to make the predicate condition is satisfied.

Thus, it can be seen from Table 6 that USE reduces 21.47% in the ATM model, and in the INRES model, the USE decreases by 24.20% in the comparison of average path feasibility value in aspect of the transition path feasibility values. Which indicate that FMM has a good effect, making the transition path length and the type of guard more trade-off between, so that the feasible transition path is more likely to be triggered in the late of generating test data.

After the experimental and analysis of the results of (Q1) and (Q2), the MBFS-FTP method proposed in this paper for automatic generation feasible transition paths has been valid in this experiment, both in terms of validly and efficiency. Better results, CCM and FMM are met our test requirements, in the future we will find more EFSM models to verify the effectiveness of this method to make it more practical.

6. Conclusions

Because of the existence of the infeasible path, which makes the test of EFSM-based model is still a challenging problem till now. In this paper, we propose a method (MBFS-FTP) for automatic generation of feasible transition paths based on EFSM model. We take advantage of the properties of BFS to shorten the path and improve the path node mark for designing MBFS. Then CCM is designed and using FMM, a feasible transition path set with short path is obtained, which lays the foundation of test data generation in later. Firstly, because the EFSM model is a static model, which cannot be driven implementation, this paper uses UML modeling method, which using SMC tool to build the executable model of EFSM, through initializing the value of the input variable to drive EFSM model dynamic execution. In order to avoid the problem of state explosion, and save the time of converting state path to transition path, we design the directed graph conversion method for converting state diagram to transition diagram, and then, we use MBFS to generate

transition path automatically. In the process of automatically generating transition paths, CCM is designed to avoid the conflicting transition existing in the transition path, therefore, every transition path generated for the target transition is executable. Aiming at multiple paths for the target transition, FMM is designed, which consider the length of the executable path, the type of operation and the type of guard, measure the feasibility of transition path, and select the transition path with high feasible value into the executable transition path set. Finally, a feasible transition path set with shortest length is obtained. After the verification of two EFSM models (ATM and INRES), our method proposed in this paper has high performance for automatic generation of feasible transition paths.

Now the research achievement of this paper is still a preliminary result, there are still many shortages need to be improved, further research mainly from the following several aspects: 1) Further studying effective method for automatic generation of feasible transition path based on EFSM model, and how to improve intelligent optimization algorithm for more effective automatic generation of test data; 2) Finding a suitable and efficient method to make the technology of automatically generate test cases based on EFSM model instrument, which applied to the actual situation and improve software testing efficiency in model testing.

Funding

This research was supported in part by the National Natural Science Foundation of China (Grant No.61502430), Science Foundation of Zhejiang Sci-Tech University (ZSTU) (Grant No.21062291-Y).

References

[1] Beizer, B. (2003). *Software Testing Techniques*. Dreamtech Press.

[2] Cohen, M. B., (2019). The Maturation of search-based software testing: Successes and challenges. *Proceedings of 2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)*(pp. 13-14), Montreal, QC, Canada, 2019.

[3] Cao, Z., Wang, Y., Guo, P. & Tian, B., (2020). EFSM Test data generation based on fault propagation and multi-population genetic algorithm. *Proceedings of 2020 7th International Conference on Dependable Systems and Their Applications (DSA)*(pp. 240-245), Xi'an, China, 2020.

[4] Sajjad, M., Wasim, M., Shahbaz, M., Saghar, K. & Khan, U. G., (2018). Dynamic testing of C program interfaces based on FSM modeling. *Proceedings of 2018 International Conference on Frontiers of Information Technology (FIT)*(pp. 24-29), Islamabad, Pakistan, 2018.

[5] Nguena, T. O., Prestat, D. & Avellaneda, F., (2019). Fault detection in timed FSM with timeouts by SAT-solving. *Proceedings of 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*(pp. 326-333), Sofia, Bulgaria, 2019.

[6] He, C., Cui, A. & Chang, C. H., (2019). Identification of state registers of FSM through full scan by data analytics. *Proceedings of 2019 Asian Hardware Oriented Security and Trust Symposium (Asian HOST)*(pp. 1-6), Xi'an, China, 2019.

[7] Hedley, D., & Hennell, M. A. (1985, August). The causes and effects of infeasible paths in computer programs. *Proceedings of the 8th international conference on Software engineering* (pp. 259-266). IEEE Computer Society Press.

[8] Turlea, A., (2019). Search-based testing using EFSMs. *Proceedings of 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*(pp.100-103), Berlin, Germany, 2019..

[9] Turlea, A., Ipate, F. & Lefticaru, R., (2017). A test suite generation approach based on EFSMs using a multi-objective genetic algorithm. In *2017 19th International Symposium on Symbolic and Numeric*

Algorithms for Scientific Computing (SYNASC)(pp.153-160), Timisoara, Romania, 2017.

[10] Turlea, A., Ipate, F. & Lefticaru, R., (2018). Generating complex paths for testing from an EFSM. *Proceedings of 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*(pp. 242-249), Lisbon, Portugal, 2018.

[11] Ural, H., & Yang, B. (1991). A test sequence selection method for protocol testing. *IEEE Transactions on Communications, 39(4)*, 514-523.

[12] Miller, R. E., & Paul, S. (1992). Generating conformance test sequences for combined control and data flow of communication protocols. In *Protocol Specification, Testing and Verification, XII* (pp. 13-27).

[13] Kalaji, A. (2010). *Search-based software engineering: A search-based approach for testing from extended finite state machine (EFSM) models* (Doctoral dissertation, Brunel University, School of Information Systems, Computing and Mathematics).

[14] Huang, C. M., Jang, M. Y., & Lin, Y. C. (1999). Executable EFSM-based data flow and control flow protocol test sequence generation using reachability analysis. *Journal of the Chinese Institute of Engineers, 22(5)*, 593-615.

[15] Hierons, R. M. (2004). Testing from a nondeterministic finite state machine using adaptive state counting. *IEEE Transactions on Computers, 53(10)*, 1330-1342.

[16] Zhao, B., Chen, B., & Qu, Y. (2007). An improved transformation perform analysis test sequence generation algorithm. *Journal of China University of Science and Technology, 37 (9)*, 1096-1100.

[17] Yang, R., Chen, Z., & Zhang, Z. (2014). An automation test case generation method based on extend finite state machine. *Chinese Science, Information Science, 5(003)*.

[18] Yang, R., Chen, Z., Xu, B., Zhang, Z., & Zhou, W. (2012). A new approach to evaluate path feasibility and coverage ratio of EFSM based on multi-objective optimization. In *SEKE* (pp. 470-475).

[19] Shu, T., Ye, T., Yin, X., & Xia, J. (2016). A test generation method for efsm-based protocols using the transitions feasibility estimation. *International Journal of Control and Automation, 9(5)*, 207-218.

[20] Wang, W., Zhao, R., Shang, Y., & Liu, Y. (2016, October). Test data generation efficiency prediction model for EFSM based on MGGP. *Proceedings of International Symposium on Search Based Software Engineering* (pp. 176-191). Springer, Cham.

[21] Korel, B., Singh, I., Tahat, L., & Vaysburg, B. (2003). Slicing of state-based models. In *Software Maintenance, 2003. ICSM 2003. Proceedings of International Conference on* (pp. 34-43). IEEE.

[22] Abdurazik, A., & Offutt, J. (2000, October). Using UML collaboration diagrams for static checking and test generation. *Proceedings of International Conference on the Unified Modeling Language* (pp. 383-395). Springer, Berlin, Heidelberg.

[23] Yano, T., Martins, E., & de Sousa, F. L. (2011, March). MOST: A multi-objective search-based testing from EFSM. In *Software Testing, Verification and Validation Workshops (ICSTW), Proceedings of 2011 IEEE Fourth International Conference on* (pp. 164-173). IEEE.

Biao Wu was born in 1989. He received the B.S. from Hubei University of Technology in 2013 and the M.E. from Zhejiang Sci-Tech University in 2016 both in the People's Republic of China. He received the Ph.D. from Yamaguchi University, Japan in 2021. Since April of 2021, he has been a Lecturer at Zhejiang Sci-Tech University. His main research interests include software testing and program net theory.

Qi-Wei Ge received the B.E. from Fudan University, China, in 1983, M.E. and Ph.D. from Hiroshima University, Japan, in 1987 and 1991, respectively. He was with Fujitsu Ten Limited from 1991 to 1993. He was an Associate Professor at Yamaguchi University, Japan, from 1993 to 2004. Since April of 2004, he has been a Professor at Yamaguchi University, Japan. His research interest includes Petri nets, program net theory and combination.