FSM BASED RE-TESTING METHODS

Khaled El-Fakih+, Nina Yevtushenko++ and Gregor v. Bochmann+++
+Department of Computer Science, American University of Sharjah, UAE, kelfakih@aus.ac.ae
++Tomsk State University, 36 Lenin Str., Tomsk, 634050, Russia, yevtushenko.rff@elefot.tsu.ru
+++ School of Information Technology and Engineering, University of Ottawa, Canada, bochmann@site.uottawa.ca

Abstract The selection of appropriate test cases is an important issue in software engineering. A number of methods are known for the selection of a test suite based on the specification and an implementation under test given in the form of a finite state machine (FSM). In realistic applications, this specification evolves incrementally throughout incorporating frequent modifications. In this paper, we adapt three well-known test derivation methods, namely the W, Wp, and HIS methods, for generating tests that would test the modified parts of the evolving specification. Application examples are provided.

Keywords: Conformance testing, finite state machines, test derivation, re-testing

1. INTRODUCTION

Many methods have been developed for deriving tests for a system represented by a Finite State Machine (FSM) model. The purpose of these tests is to determine whether an implementation of the system conforms to (i.e., is correct with respect to) its specification. Usually a conforming implementation is required to have the same I/O behavior.

In realistic applications, maintaining a system modeled by a given specification machine involves modifying its specification as a result of changes in the users' requirements and designers implement incrementally these modifications. Testing the whole system implementation after each modification is considered expensive and time consuming. Therefore, it is important to generate tests (called re-tests) that would only test the modified

parts of the implementation that correspond to the modified parts of its specification. This would reduce the maintenance cost of such a system, which is about two-thirds of the cost of the software production [10].

In this paper, we present test generation methods (called henceforth as re-testing methods) that select tests (called re-tests) for testing the modified parts of the system specification, in order to check that these modifications were correctly implemented in the system implementation. Here we assume that the parts of the system implementation that correspond to the unmodified parts of the system specification are left intact. Moreover, we also reasonably assume that before modifying the system specification, its implementation was tested and found conforming to this specification. These methods are based on well-known test derivation methods called the W[1], Wp [3], and HIS [8] methods.

The problem of deriving re-testing sequences can be converted into the problem of test derivation from an FSM with a fault function [7] or from its generalization [5]. In this case, potential implementations are represented as all complete sub-machines of a given nondeterministic FSM that is called a mutation machine. However, the mutation machine from which re-testing sequences are derived, is special. Each unmodified specification transition is a deterministic transition of the mutation machine while each modified transition becomes chaotic; each pair (state, output) becomes possible as its tail state and output in a potential implementation. In other words, on one hand, the mutation machine has a number of deterministic transitions that can be used deriving a test suite, while on the other hand, a number of all possible paths that include modified transitions becomes exponential. Based on these features, we propose a proper technique for test derivation. First, we do not explicitly enumerate all possible implementation paths under an appropriate input sequence and secondly, we essentially use unmodified specification transitions that still remain deterministic in the mutation FSM. However, we always mention when our method delivers the same test suite as the methods for test derivation from an arbitrary mutation machine proposed in [7], [5].

This paper is organized as follows. Section 2, describes the finite state machine model, and Section 3 briefly describes the W, Wp, and HIS methods for generating tests from a given FSM specification. Based on these methods, our re-testing methods are presented in Section 4 with appropriate application examples. Section 5 concludes the paper.

2. FINITE STATE MACHINES

A deterministic *finite state machine* is a 7-tuple $M = (S, X, Y, \delta,$

λ, D_M, s_1), where: S is a finite set of states, s_1 is the *initial state*, X is a finite set of input symbols, Y is a finite set of output symbols, δ is a next state (or transition) function:δ: D_M --> S, λ is an output function: λ: D_M --> Y, and D_M is a specification domain: $D_M \subseteq S \times X$.

We use as in [3] the notation "(s_i -x/y-> s_j)" to indicate that the FSM M at state s_i responds with an output y and makes the transition to the state s_j when the input x is applied. State s_i is said to be the *head* or *starting* state of the transition, while s_j is said to be the *tail* or *ending* state of the transition. If we are not interested in the output we write "s_i-v->s_j" when an input sequence v is applied at state s_i. FSM M is said to be *completely specified* or simply a *complete* FSM, if $D_M = S \times X$; otherwise, M is said to be *partially specified* or simply a *partial* FSM. In the complete FSM, we omit the specification domain D_M, i.e. a complete FSM is a 6-tuple $M = (S, X, Y, \delta, \lambda, s_1)$. The concatenation of sequences v_1 and v_2 is the sequence $v_1.v_2$. For a given alphabet Z, Z^* is used to denote the set of all finite words over Z. Let V be a set of words over alphabet Z. The prefix closure of V, written ***Pref***(V), consists of all the prefixes of each word in V, i.e. ***Pref***$(V) = \{\alpha \mid \exists \gamma\ (\alpha.\gamma \in V)\}$. The set V is *prefix-closed* if ***Pref***$(V) = V$.

Let $M_S = (S, X, Y, \delta_S, \lambda_S, D_S, s_1)$ and $M_I = (T, X, Y, \Delta_I, \Lambda_I, D_I, t_1)$ be two FSMs. In the following sections M_S usually represents a protocol specification while M_I denotes an implementation, and thus, FSM M_I is further assumed to be complete. Given an input sequence $\alpha = x_1\ x_2\ ..\ x_k \in X^*$, α is called a *defined input sequence*(**DIS**) at state $s_i \in S$, if there exist k states $s_{i1}, s_{i2}, ..., s_{ik} \in S$ such that there is a sequence of specified transitions s_i- x_1-> s_{i1} --> ... --> $s_{i(k-1)}$- x_k-> s_{ik} in the finite state machine M_S. Hereafter, **DIS**($M_S|s_i$) will be used to denote the set of all the defined input sequences at state s_i of machine M_S.

We say that states s_i of M_S and t_j of M_I are compatible if **DIS**($M_S|s_i$) $\cap$ **DIS**($M_I|t_j$)=$\varnothing$ or if $\forall\ \alpha \in$ **DIS**($M_S|s_i$) $\cap$ **DIS**($M_I|t_j$) it holds that $\lambda_S(s_i, \alpha) = \Lambda_I(t_j, \alpha)$. Otherwise; we say that states s_i and s_j are distinguishable. Input sequence $\alpha \in$ **DIS**($M_S|s_i$) $\cap$ **DIS**($M_I|t_j$) such that $\lambda_S(s_i, \alpha) \neq \Lambda_I(t_j, \alpha)$ is said to distinguish the states s_i and t_j. An FSM is said to be *reduced* if its states are pair-wise distinguishable. If the FSMs happen to be complete, then the definition of compatible states reduces to the definition of equivalent states (see for example, [4]).

3. REVIEW OF THE W, WP, AND HIS METHODS

In the following section we briefly describe test derivation methods where the specification is given as a reduced FSM M_S while an implementation under test (IUT) is modeled by a complete FSM M_I.

Let t_i be a state of M_I and s_j be a state of M_S. Consider set V of input sequences such that $V \subseteq \mathbf{DIS}(M_S|s_j)$. State t_i is said to be equivalent to s_j with respect to the set V (written as $t_i \cong_V s_j$), if $\Lambda_I(t_i, \alpha) = \lambda_S(s_j, \alpha)$ holds for any $\alpha \in V$. In other words, for each input sequence of V, a behavior of M_I at state t_i coincides with that of M_S at state s_j.

We say that M_I *conforms* to M_S if and only if $t_1 \cong_{\mathbf{DIS}(M_S|s_1)} s_1$, where t_1 and s_1 are the initial states of M_I and M_S, respectively. In other words, for each input sequence where a behavior of M_S is defined, M_I has the same behavior, i.e. the implementation is quasi-equivalent to the specification [4]. This conformance relation corresponds to the notion of weak conformance [9].

A set $\boldsymbol{Q}$ of input sequences is called a *state cover set* of FSM M_S if for each state s_i of S, there is an input sequence $\alpha_i \in Q$ such that $s_1\text{-}\alpha_i\text{->}s_i$.

Usually the testing methods reviewed in this section use state identification facilities in order to check that each state and each transition defined in the specification also exists in the implementation. These facilities have certain input/output behaviors that can distinguish the states of an FSM. Given a reduced FSM M_S and a state $s_i \in S$, a set $W_i \subseteq \mathbf{DIS}(M_S|s_i)$ of defined input sequences at state s_i is called a *state identifier* of state s_i if for any other state s_j there exists $\alpha \in W_i \cap \mathbf{DIS}(M_S|s_j)$ such that $\lambda_S(s_i, \alpha) \neq \lambda_S(s_j, \alpha)$. We now define a collection of state identifiers that has been named a family of *harmonized identifiers* [6], [8] or a separating family [11]. A *separating family* is a collection of state identifiers W_i, $s_i \in S$, which satisfy the condition that for any two states s_i, and s_j, $i \neq j$, there exist $\beta \in W_i$ and $\gamma \in W_j$ which have common prefix α such that $\alpha \in \mathbf{DIS}(M_S|s_i) \cap \mathbf{DIS}(M_S|s_j)$, and $\lambda_S(s_i, \alpha) \neq \lambda_S(s_j, \alpha)$. A separating family exists for any reduced (partial or complete) machine.

A *characterization set* of the FSM M_S, often simply called a *W set*, is a set of input sequences which satisfies the following conditions:

(1) For any $s_k \in S$, $W \subseteq \mathbf{DIS}(M_S|s_k)$,

(2) For any two states s_i, and s_j, $i \neq j$, there exists $\beta \in W$ such that $\lambda_S(s_i, \beta) \neq \lambda_S(s_j, \beta)$.

A W set always exists for a reduced completely specified machine. However, the W set does not always exist for a reduced partially specified machine.

Given a specification reduced FSM $M_S = (S, X, Y, \delta_S, \lambda_S, D_S, s_1)$, $|S|=\boldsymbol{n}$, and a complete implementation FSM $M_I = (T, X, Y, \Delta_I, \Lambda_I, t_1)$ such that $|T|=\boldsymbol{n}$, let W be a characterization set of M_S (if exists) and $F = \{W_1, \ldots, W_n\}$ be a separating family of M_S.

All the methods have two phases. In the first so-called *state identification* phase, they establish a one-to-one mapping $h_{S\text{-}I}$: $S \rightarrow T$ by the use of a characterization set W or a separating family F. Given a prefix-closed state

cover set $Q = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of the specification FSM, for each state $s_j \in S$, the state identification phase comprises the sequences: $r.\alpha_j.W_j$ (HIS method) or $r.\alpha_j.W$ (W and Wp methods).

We note that, if the specification FSM M_S is partial, a characterization set W may not exist; in this case, the W and Wp methods cannot be applied. However, the HIS method can be applied.

If FSM M_I passes the state identification test sequences, then there exists one-to-one mapping $h_{S\text{-}I}$: $S \rightarrow T$ such that: $h(s_j) = t \Leftrightarrow s_j \cong_{Wj} t$ in the HIS method, and $h(s_j) = t \Leftrightarrow s_j \cong_{W} t$ in the W and Wp methods.

The second so-called *transition testing* phase, assures that for each state $s \in S$, and input $x \in X$ that is defined at state s the mapping $h_{S\text{-}I}$ satisfies the following property:

$$\lambda_S(s, x) = \Lambda_I(h_{S\text{-}I}(s), x) \text{ and } h_{S\text{-}I}(\delta_S(s, x)) = \Delta_I(h_{S\text{-}I}(s), x) \qquad \textbf{(X-1)}$$

For this purpose, for each sequence $\alpha_j \in Q$ that takes the specification FSM to appropriate state s_j, and each $x \in X$ that takes the M_S from state s_j to state s_k, the testing transition phase includes the set of sequences:

$r.\alpha_j.x.W_k$ in the HIS and Wp methods, where W_k is a state identifier of the state s_k in the specification FSM (for the Wp method, we have $W_k \subseteq W$) or

$r.\alpha_j.x.W$ in the W method

If FSM M_I passes the test sequences of both testing phases, then it is quasi-equivalent to the specification FSM, i.e. is a conforming implementation. If the specification FSM is complete then the quasi-equivalence relation reduces to the equivalence relation, i.e. the specification FSM and its conforming implementation have the same Input/Output behavior.

4. FSM BASED RE-TESTING

4.1 Problem Definition

Let the reduced FSM $M_S = (S, X, Y, \delta_S, \lambda_S, D_S, s_1)$ be the specification of a given system. We assume that the complete implementation FSM $M_I = (T, X, Y, \Delta_I, \Lambda_I, t_1)$ of M_S with the same number of states has been tested and found conforming to M_S. Therefore, there exists a one-to-one mapping $h_{S\text{-}I}$: $S \rightarrow T$ such that for each state $s \in S$ and input $x \in X$ that is defined at state s, (X-1) holds.

Let the reduced $M_S' = (S, X, Y, \delta_S, \lambda_S, D_S', s_1)$ be the modified specification, and $M_I' = (T, X, Y, \Delta_I, \Lambda_I, t_1)$ be the modified implementation that must conform to M_S'. We assume that only transitions corresponding to the modified parts of M_S' have been changed in M_I' and we want to generate test sequences for the modified parts of the system specification, in order to

check that these modifications were implemented correctly in the modified implementation M_I'. In other words, for each unmodified transition (s_j-x/y->s_k) of the M_S', we assume that transition ($h_{S\text{-}I}(s_j)$-x/y->$h_{S\text{-}I}(s_k)$) has not been changed in the modified implementation M_I'. We note that in case where new states are added (or deleted) to (or from) M_S, we let S' denote the set of states of M_S' and T' denote the set of states of M_I', respectively.

In general, we have the following types of modifications that can be made in M_S and implemented by a designer in M_I:

(1) outputs of some transitions are modified, (2) tail states of some transitions are modified, (3) outputs and tail states of some transitions are modified, (4) new transitions are added (5) some transitions are deleted, (6) new states are added, and (7) some states are deleted.

4.2 The Re-testing Methods

The re-testing methods adapted for the W, Wp, and HIS test derivation methods have also two phases. In the first phase, re-tests are selected in order to check (or re-identify) some states of the modified specification in the new implementation, and in the second re-testing phase, re-tests are selected to check each modified transition for correct output and tail state. Here, we examine different cases that can be used to generate short re-testing test sequences.

For convenience, hereafter, we use the input symbols a and b for unmodified transitions, and x and z for modified ones.

4.3 Case-1: The Unmodified Part of the Modified Specification is Reduced

Here we assume that the unmodified part $UP\text{-}M_S'$ of the modified specification M_S' is reduced. Then, there exist state identifiers $W_1,\ldots, W_n$, which satisfy the following conditions: 1) each W_i is a subset of the defined input sequences at state $s_i \in S$ in the $UP\text{-}M_S'$, 2) given two states s_i, and s_j, $i \neq j$, there exist sequences in W_i and W_j with the common prefix β such that $\lambda_S(s_i, \beta) \neq \lambda_S(s_j, \beta)$.

We note that since the unmodified part of the specification can be partial, a characterization set W may not exist. However, we can always select state identifiers with the above conditions.

4.3.1 General solution

For each modified edge (s_j-x/y->s_k), its corresponding re-testing test cases are formed as follows:

If we use the HIS or Wp methods: $r.\alpha_j.x.W_k$, **(1-a)**
where W_k is a state identifier of state s_k.
If a characterization set W exists and we use the W method: $r.\alpha_j.x.W$ **(1-b)**

Theorem 1. Given a modified specification M_S' and its implementation M_I', let the unmodified part of M_S' be reduced and have state identifiers W_1, …, W_n. If implementation M_I' passes the re-testing test suite which consists of the union of the test cases over all modified transitions as given in Formulae (1-a) or (1-b), then the implementation M_I' is quasi-equivalent to M_S'.

We omit the proof of Theorem 1 since it is a particular case of Theorem 2.

We note that if the specification FSM is complete then the above case is a particular case of the advanced procedure in [7] since each state identifier is a so-called stable state identifier, i.e. is a state identifier of the corresponding state in each potential implementation.

4.3.2 An Optimized Solution

We observe that a sequence that distinguishes two states of the initial specification and traverses only unmodified transitions when applied at these states, also distinguishes the corresponding states of the modified implementation M_I'. Moreover, when the unmodified part of the modified specification is reduced, the mapping $h_{S\text{-}I}(s)$: $S \rightarrow T$ between the initial specification and its conforming implementation is the only candidate that can satisfy (X-1) for the modified specification and its implementation. Therefore, we do not need to re-identify states of the modified implementation. Moreover, the test suite constructed using the formulae of the general solution may be shortened if we use, when checking certain transitions, shorter state identifiers that pass through already tested transitions rather than those generated only from the unmodified part of the modified specification. In other words, instead of using state identifiers derived in advance, we can generate shorter state identifiers, as modified transitions are tested. For this purpose, we assume that a linear order "<" over modified transitions of the specification is given. This order satisfies the following property: If $\alpha_r.z \in Q$ is a prefix of $\alpha_j \in Q$, then for any two modified transitions (s_r-z->s_l) and (s_j-x->s_k), transition (s_r-z->s_l)<(s_j-x->s_k). In this case, when checking a modified transition, we can use lower order transitions (or already checked transitions) to generate shorter re-testing sequences. The reason is that if the implementation at hand passes retesting sequences for transition (s_r-z/y->s_l) then it has the corresponding transition ($h_{S\text{-}I}(s_r)$ -z/y-> $h_{S\text{-}I}(s_l)$), and this transition can be used for retesting a higher order transition. In this section, we illustrate by an example the advantage of

using such a linear order. However, we do not discuss how to derive an order that provides the shortest re-testing sequences.

For each modified edge (s_j-x->s_k), its corresponding re-testing test cases are formed as in Formulae (1-a) or (1-b). However, as a state identifier of state s_k, we use state identifier W_k (or characterization set W in W method) of the part of M_S that comprises unmodified transitions or modified transitions (s_r-z->s_l) where (s_r-z->s_l) < (s_j-x->s_k). In other words, for testing transition (s_j-x->s_k) the state identifier W_k has sequences that if applied at state s_k only traverse unmodified transitions or modified transitions (s_r-z->s_l) < (s_j-x->s_k). This allows, when checking a modified transition, the use of already re-checked transitions (or lower order transitions) in order to generate shorter state identifiers.

Theorem 2. Given a modified specification M_S' and its implementation M_I', let the unmodified part of M_S' be reduced, and for each modified transition (s_j-x->s_k) W_k is a state identifier of state s_k in the part of M_S that comprises unmodified transitions or modified transitions (s_r-z->s_l) < (s_j-z->s_k). If implementation M_I' passes the re-testing test suite which consists of the union of the test cases over all modified transitions as given in Formulae (1-a) or (1-b), then the implementation is quasi-equivalent to M_S'.

Proof. A proof of Theorem 2 is given in [2].

As an example for the general and optimized solution methods based on the HIS method, we consider the modified specification FSM M_1 shown in Fig. 1. FSM M_1 has the input set X={*a, b, c*}, output set Y ={y_1, y_2, y_3}. The labels of the modified transitions are shown in bold. The unmodified part of FSM M_1 has a separating family {w_1, w_2, w_3, w_4} of state identifiers, where w_1= {*bb*}, w_2= {*bb*}, w_3= {*bb, c*} and w_4= {*bb, c*}. In fact, for each state s_i of M_1 we have the following input/output sequences in response to w_i.

	s_1	s_2	s_3	s_4
b b	y_1 y_2	y_2 y_2	y_2 y_1	y_2 y_1
c			y_2	y_1

Table 1. Responses of M_1 to state identifiers (if defined)

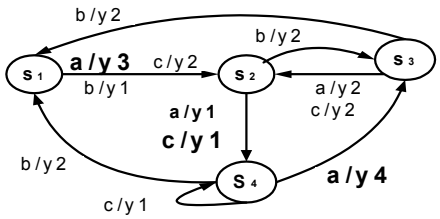

Figure 1. *Specification* M_1

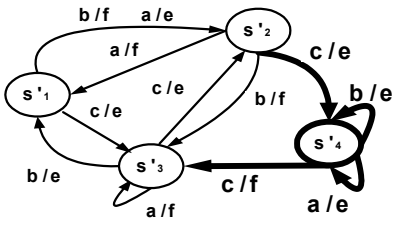

Fig. 2. *Specification* M_S'

None of the above state identifiers passes through a modified transition if applied at an appropriate state, and thus, the unmodified part of M_S' is

reduced. According to the general solution, in order to re-test the modified transitions, the following re-testing sequences of length 24 are generated using Formula (1-a): $\{r.\alpha 1.a.w2+r.\alpha 2.c.w4+r.\alpha 4.a.w3\} = \{r.\varepsilon.a.bb, r.b.c.bb, r.b.c.c, r.ba.a.bb, r.ba.a.c\}$, where $\alpha 1= \varepsilon$, $\alpha 2= b$ and $\alpha 4= ba$ are the corresponding sequences of the state cover set .

We now use the following linear order over modified transitions ($s1$-a->$s2$) < ($s4$ a->$s3$) < ($s2$-c->$s4$). In order to re-test the modified transition ($s1$-a-> $s2$), the re-testing test sequence $\{r.a.w2=r.a.bb\}$ is used. If the implementation at hand passes this sequence, then we have the following responses to the input a :

At state $h_{S\text{-}I}(s1)$: $y3$, at state $h_{S\text{-}I}(s2)$: $y1$, and at state $h_{S\text{-}I}(s3)$: $y2$,

Consequently, in order to re-test transition ($s4$-a->$s3$), the re-testing test sequence $r.\alpha 4.a.a = r.b.a.a.a$ will be enough instead of $r.\alpha 4.a.w3$, since $r.\alpha 4$ reaches state $s4$ through unmodified transitions and if afterwards, the modified implementation produces the expected output $y4$ to the input symbol a, then a becomes a distinguishing sequence for the part of the specification comprising unmodified transitions and transitions ($s1$-a->$s2$) and ($s4$-a->$s3$), where for the last transition only its output has been checked; therefore, a is a distinguishing sequence for the corresponding part of the modified implementation.

Afterwards, in order to check the ending state $s4$ of the modified transition ($s2$-c->$s4$), the distinguishing sequence a can be used instead of the previous state identifier $w4= \{bb, c\}$. Hence, the re-testing test sequence of this transition is $r.\alpha 2.c.a = r.b.c.a$. Therefore, according to the optimized solution method, in order to re-test the modified transitions, the following re-testing sequences are generated using the above linear order and Formulae (1-a): $\{r.a.bb + r.b.a.a.a + r.b.c.a\}$.

The total length of these sequences is 13, where the total length of those generated using Formula (1-a) of the general solution method is 24. The HIS method generates a test suite of length 51 if the whole specification of M_S' is considered for test derivation.

4.4 Case-2: Each State of the Modified Specification is Reachable Through Unmodified Transitions and the Unmodified Part is Not Reduced.

In some cases, the unmodified part of the modified specification M_S' is not reduced. However, each state of M_S' is reachable through some unmodified transitions. Since each state of M_S' can be reached through unmodified transitions, the only possible correct mapping between the states of M_S' and M_I' is the old mapping established between the states of M_S and M_I. Therefore, in order to check that this mapping still holds for the states of

the modified specification and implementation the only states that have state identifiers passing through modified transitions have to be re-identified in the new implementation. Moreover, in order to re-identify such a state, it is enough to apply only the sequences of the corresponding state identifier that pass through modified transitions.

Let Q be a prefix-closed state cover set such that its sequences do not traverse modified transitions if applied at the initial state of M_S'. Let also $F = \{W_1, \ldots, W_n\}$ be a separating family of the modified specification, and W be a characterization set (if exists).

i) State re-identification phase

For each state s_r such that some sequences of W_r traverse modified transitions if applied at s_r the state re-identification sequences are formed as follows:

$$r.\alpha_r.W_r' \qquad \textbf{(2-1a)}$$

where $W_r' \subseteq W_r$ (or $W_r' \subseteq W$ for the W and Wp methods) comprises each sequence of the state identifier W_r (of the characterization set W) that, if applied at state s_r of the modified specification, traverses a modified transition. We note that each state of M_S' for which all sequences of the state identifier traverse only unmodified transitions, does not need to be re-identified.

ii) Re-testing modified transitions phase

For each modified edge (s_j-x->s_k), its corresponding re-testing test sequences are formed as shown in Formulae (1-a) and (1-b).

Theorem 3. Given the modified specification M_S' and its implementation M_I', let $F = \{W_1, \ldots, W_n\}$ be a separating family of M_S' and W be a characterization set of M_S' (if exists). Let also Q be a prefix-closed state cover set of M_S' such that each sequence of the set Q does not traverse a modified transition if applied at the initial state. If implementation M_I' passes the re-testing test suite which is the union of the re-testing test sequences given in Formula (2-1a) and Formulae (1-a) or (1-b), then the implementation is quasi-equivalent to M_S'.

We omit the proof of Theorem 3 since it is a particular case of Theorem 4.

We note that union of the test cases given in Formula (2-1a) and Formula (1-a) coincides with the test suite returned by procedures in [7], [5], since both methods return a test suite without input sequences that traverse only unmodified transitions. However, here we underline the advantage of selecting a state cover set and state identifiers with sequences that do not traverse any modified transition. In this case, the old image of a corresponding state of a modified specification must be preserved and therefore, there is no need to check unmodified transitions at the corresponding state.

4.5 Case-3: Some States are only Reachable Through Modified Transitions

In some cases, the unmodified part of the modified specification M_S' is not reduced and some states of M_S' are only reachable through modified transitions. This case always holds when additional states are introduced when modifying the specification. Here, for the subset of states, say $S_{r\text{-}m}$ of the modified specification that are only reachable through modified transitions, the old conforming mapping might not be preserved between the new specification and its implementation, i.e. some $s_k \in S_{r\text{-}m}$ of the modified specification might be mapped to a new state of its implementation (say $t_l \in T_{r\text{-}m}$), different from t_k. Each such state must be re-identified in the new implementation and moreover, differently from former two cases, we have to check unmodified transitions from this state.

As an example, we modify the specification M_S shown in the upper part of Fig. 3 and obtain the FSM M_S' shown in the upper of Fig. 4. The modified transitions are shown as bold lines. We note that the mapping between states of M_S and its conforming implementation M_I, shown in the lower part of Fig. 3, is $h_{S\text{-}I}(s_k) = t_k$ for k= 1,...,4. Moreover, we let M_I', shown in the lower part of Fig. 4, be the implementation of M_S'. M_S' has $W = \{aa\}$ as a characterization set which is a state identifier of each state. In fact, we have the following output responses to aa. For state s_1, xx. For state s_2, xy. For state s_3, yy, and for state s_4, yx.

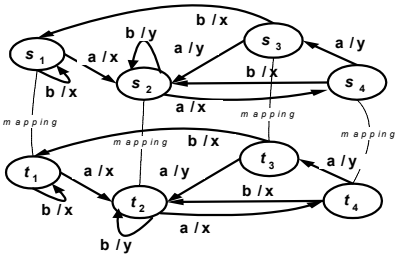

Figure 3. M_S *and* M_I

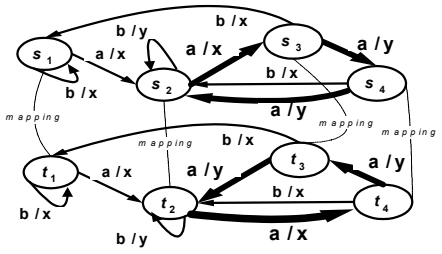

Figure 4. M_S' *and* M_I'

We note that the modified implementation state t_4 of M_I' has the output response yy to aa, i.e. state s_3 of M_S' is W-equivalent to t_4 of M_I' while state s_4 of M_I' is W-equivalent to t_3 of M_I'. States t_1 and t_2 of M_I' are W-equivalent to s_1 and s_2 of M_I'. The mapping is preserved for the modified transitions from states t_3 and t_4 under input a. However, it is not preserved for the unmodified transitions at states s_3 of M_S' and s_4 of M_S' under input b, i.e. M_I' is a wrong implementation of M_S'. Therefore, appropriate unmodified transitions also need to be checked in order to kill such a wrong mapping.

As in the previous section, we select a prefix-closed state cover set with the following property. Given state $s_j \in S$ of M_S' reachable through unmodified transitions, we select the sequence $\alpha_j \in Q$ that does not traverse any modified transition.

The re-testing method has two phases. In the first phase, some states of the modified specification are re-identified in the new implementation. It may occur that for some states of the modified specification that are only reachable through modified transitions their old image must still be preserved in the new implementation. In particular, those are the states that have a so-called stable identifier [7] that distinguishes a state from any other state in each possible implementation. For each such state, we derive a state identifier (if it exists) that kills, through the re-identification phase, implementations where the state has a new image. We start from the set of states that are reachable through unmodified transitions. As in Case-2, the old mapping must still be valid for these states and only modified transitions from these states need to be checked. Moreover, given such a state, if the sequences of its state identifier do not traverse modified transitions then the state does not need to be re-identified. Otherwise, we select state re-identification sequences using Formula (2-1a) of Case-2, where in order to check the new mapping of a state, say s_j, we concatenate the sequence $r.\alpha_j$ with each sequence of the state identifier of the s_j that passes through a modified transition. Then, we iteratively identify all other states for which the old mapping must be preserved; however, their state identifiers are derived in a proper way as described below. Afterwards, since each remaining state, say $s_j \in S_{r\text{-}m}$, of the modified specification could have a new image, i.e. s_j could be mapped to say $t_k \in T_{r\text{-}m}$ instead of the old image $h_{S\text{-}I}(s_j) = t_j$, re-tests are selected to re-identify the image of s_j in the new implementation, i.e. check (or establish) that s_j is W-equivalent to t_k, and to check that this mapping is conforming. In order to re-identify s_j, re-tests are selected by concatenating $r.\alpha_j$ with each sequence in the state identifier of state s_j including sequences which traverse only unmodified transitions Moreover, in order to check that this mapping is a valid one (i.e. kill wrong mappings), re-tests are selected to check each outgoing transition from s_j for correct output and ending state in the new implementation.

In order to implement the above steps, we determine a subset S_u of the set of states of the modified specification such that for the states in S_u the old mapping between the states of the modified specification and its conforming modified implementation must still be preserved. The set S_u enjoys a nice property. For each state in S_u, we do not need to check its outgoing unmodified transitions. In the following paragraph, we determine which states may be in the set S_u and derive the set S_u together with a separating family $F=\{W_1, \ldots, W_n\}$ (or characterization set W) so that if the

implementation passes the re-identification test sequences, then there exists one-to-one mapping h: $S \rightarrow T$ such that the following property holds.

For each state $s_i \in S_u$ we have: $s_i \cong_{w_i} t \Leftrightarrow t = h_{S\text{-}I}(s_i)$. **(X-2)**

First, we add to the empty set S_u each state s_j that is reachable from the initial state through unmodified transitions. As in Case-2, the images of these states have to be still preserved in the new implementation. Then, for state s_j and each state $s_i \in S$, $s_i \neq s_j$, we include in the state identifiers W_j and W_i a sequence that distinguishes the states s_j and s_i in the modified specification. We note that, as discussed for Case-2, we recommend, while building the state identifier W_j, to select the sequences that do not pass through modified transitions if applied at state s_j, since we do not need to apply these sequences while re-identifying s_j.

Afterwards, we iteratively include in S_u each state $s_j \in S \backslash S_u$, such that for each state $s_i \in S \backslash S_u$, $s_i \neq s_j$, there exists sequence β_{ij} that does not traverse modified transitions if applied at states s_i and s_j and $\lambda_S(s_i, \beta_{ij}) \neq \lambda_S(s_j, \beta_{ij})$, or there exists input x such that transitions $(s_j\text{-}x\text{->}s_k)$ and $(s_i\text{-}x\text{->}s_r)$ are unmodified, $s_k \neq s_r$ and $s_k, s_r \in S_u$. In the former case, we include sequence β_{ij} in W_i and W_j. Since β_{ij} does not traverse modified transitions if applied at states s_i and s_j we have that $\Lambda_I(h_{S\text{-}I}(s_i), \beta_{ij}) = \lambda_S(s_i, \beta_{ij})$, and $\lambda_S(s_j, \beta_{ij}) \neq \Lambda_I(h_{S\text{-}I}(s_i), \beta_{ij})$. Thus, if β_{ij} is included into W_i and W_j and the implementation passes the corresponding state re-identification sequences, then s_j is not W-equivalent to $h_{S\text{-}I}(s_i)$ (i.e $h_{S\text{-}I}(s_j) \neq h_{S\text{-}I}(s_i)$). In the latter case, we include into W_i and W_j the sequence $x\beta$ where β is a common prefix of the appropriate sequences in W_i and W_j such that $\lambda_S(s_k, \beta) \neq \lambda_S(s_r, \beta)$. Thus if $\Lambda_I(h_{S\text{-}I}(s_i), x.\beta) = \lambda_S(s_i, x.\beta)$, then $\lambda_S(s_j, x.\beta) \neq \Lambda_I(h_{S\text{-}I}(s_i), x.\beta)$. If $x.\beta$ is included in W_i and W_j and the implementation passes the corresponding state re-identification sequences, then s_j is not W-equivalent to $h_{S\text{-}I}(s_i)$. Due to the definition of state identifiers for states in S_u, such a sequence exists. If any sequence of each state identifier is defined at each state then we derive the set W as the union of all state identifiers. We note that in order to kill for s_j any mapping h_I where $h_I(s_j) \neq h_{S\text{-}I}(s_j)$, the corresponding state re-identification sequences are derived by concatenating $r.\alpha_j$ with every sequence of the set W_j (or W for the Wp and W methods).

Finally, we derive state identifiers for the remaining states in $S \backslash S_u$. For each state s_j in $S \backslash S_u$ and for each state $s_i \in S$, $s_i \neq s_j$, we include a sequence β_{ij} in W_i and W_j (if it does not already exist) such that $\lambda_S(s_i, \beta_{ij}) \neq \lambda_S(s_j, \beta_{ij})$. In order to re-identify s_j in the new implementation and kill its possible wrong images, the corresponding re-testing sequences include all re-identification sequences and re-testing sequences for testing all outgoing transitions from state s_j.

The characterization set W (for the W, and Wp methods) can be obtained as the union of state identifiers W_i, i=1, …, n (if possible). We note that in

order to reduce the number of transitions which need to be checked we use another technique than that based on stable state identifiers [7]. The main idea behind our approach is based on the observation that for each state reachable through unmodified transitions and some other states, the old image must be preserved in each conforming modified implementation. Our technique can also be used to reduce a test suite derived from a mutation machine [5] if the latter has many deterministic transitions.

i) Phase of state re-identification

For each state s_j of the modified specification that needs to be re-identified in the new implementation, we derive its state re-identification test sequences as follows:

If α_j does not traverse a modified transition, the re-identification sequences are formed as in Formula (2-1a).

If α_j traverses a modified transition then there are test sequences

$r.\alpha_j.W_j$ (HIS method); **(3-1a)**

$r.\alpha_j.W$ (W and Wp methods). **(3-1b)**

Every sequence of the set W_j (or W) must be applied after α_j, whether the sequence applied at state s_j traverses a modified transition or not.

ii) Phase of re-testing modified transitions

For each modified edge (s_j-x->s_k), where $s_j \in S_u$, its corresponding test cases are formed as in Formulae (1-a) or (1-b). For each state $s_j \notin S_u$, Formula (1-a) or (1-b) are applied for each outgoing transition from state s_j including those which are unmodified.

Theorem 4. Given the modified specification M_S' and implementation M_I', let Q be a prefix-closed state cover set of M_S' and $F = \{W_1, \ldots, W_n\}$ and W be a separating family and a characterization set (if exists) of the modified specification M_S' derived as described above. If implementation M_I' passes the re-testing test suite derived for Case-3, then the implementation is quasi-equivalent to M_S'.

Proof. As we demonstrated by the example, in Case 3, a one-to-one mapping h: $S \rightarrow T$ such that state s_i of M_S' is W_i-equivalent to state $h(s_i)$ of M_I' can be different from $h_{S\text{-}I}$. We first need to check whether the one-to-one mapping h exists at all.

We consider a relation $h \in S \times T$ such that: $(s_j, t_j) \in h \Leftrightarrow s_j \cong_{Wj} t_j$. If the implementation passes the re-identification test cases given by Formulae (2-1a), (3-1a) and (3-1b) then h is a one-to-one mapping h: $S \rightarrow T$. We next show that $h(s_j) = h_{S\text{-}I}(s_j)$ holds for each $s_j \in S_u$.

Given state $s_j \in S$ of M_S' such that the sequence $\alpha_j \in Q$ does not traverse modified transitions, if M_I' passes the test sequences given in Formulae (3-1a) then the state $h_{S\text{-}I}(s_j) = h(s_j)$. The initial state s_1 is in the set S_u, i.e. the base of induction holds.

Let us assume that $h(s) = h_{S\text{-}I}(s)$ holds for each state s of a current set S_u and that state s_j is the next state we are going to include into S_u using the procedure described above. Since h is a one-to-one mapping, for each state $s \in S_u$ it holds that $h(s_j) \neq h_{S\text{-}I}(s)$. On the other hand, for each state $s_i \in S \setminus S_u$, $i \neq j$, by construction of the state identifier, $\exists$ a sequence $\beta_{ij} \in W_j \cap W_i$ such that β_{ij} does not traverse modified transitions if applied at states s_i or s_j and $\lambda_S(s_i, \beta_{ij}) \neq \lambda_S(s_j, \beta_{ij})$, or $\exists$ a sequence $x\beta \in W_j \cap W_i$ such that the final states of unmodified transitions (s-x->s_k) and (s_i-x->s_r) are different and $\lambda_S(s_k, \beta) \neq \lambda_S(s_r, \beta)$. In the former case, M_I' has different output responses to the sequence $\beta_{ij} \in W_j \cap W_i$ at the states $h(s_j)$ and $h_{S\text{-}I}(s_i)$, i.e. $h(s_j) \neq h_{S\text{-}I}(s_i)$. In the latter case, M_I' at states $h_{S\text{-}I}(s_k)$ and $h_{S\text{-}I}(s_r)$ has different output responses to the sequence $\beta \in W_k \cap W_r$, i.e. M_I' has different output responses to the sequence $x\beta \in W_k \cap W_r$ at the states $h(s_j)$ and $h_{S\text{-}I}(s_i)$, i.e. $h(s_j) \neq h_{S\text{-}I}(s_i)$. Therefore, by induction, $h(s_j) = h_{S\text{-}I}(s_j)$ for each state $s_j \in S_u$.

For each unmodified transition (s_j-a ->s_l) from state $s_j \in S_u$ it holds that $\lambda_S(s_j, a) = \Lambda_I(h_{S\text{-}I}(s_j))$ and $h_{S\text{-}I}(s_l) = \Delta(h_{S\text{-}I}(s_j), a)$.

If M_I' passes the test cases $r.\alpha_j.x.W_k$ for a modified transition (s_j-x ->s_k) then $\lambda_S(s_j, x) = \Lambda_I(h_{S\text{-}I}(s_j), x)$ and the ending state of the transition ($h_{S\text{-}I}(s_j)$-x ->t_k) is W_k-equivalent to s_k.

Due to the construction of retesting sequences, we also check that (X-1) holds for each transition from each state $s_j \notin S_u$. Thus, if M_I' passes the test, then the mapping h satisfies (X-1), i.e. M_I' is quasi-equivalent to M_S' □

As an application example for Case-3 with the HIS method, we add to the given specification a new state $s'4$ and its corresponding incoming and outgoing edges producing the modified specification M_S' shown in Fig. 2.

The state cover set of M_S' is $Q' = \{\varepsilon, b, c, bc\}$. We consider each incoming and outgoing transition of the added state (here $s'4$) as a modified transition. Therefore, the modified transitions of M_S' are ($s'2$-c/f->$s'4$), ($s'4$-b/f->$s'4$), ($s'4$-a/e->$s'4$), and ($s'4$-c/f->$s'3$).

According to Case-3, we add to the set S_u states $s'1$, $s'2$, and $s'3$ since these states are reachable through unmodified transitions and there exists a state identifier for each of these states that does not pass through modified transitions. In this example, the sequence bb is such an identifier. In fact, we have the following input/output sequences in response to bb. For state $s'1$, ff. For state $s'2$, fe. For state $s'3$, ef, and for state $s'4$, ee.

These states, i.e., $s'1$, $s'2$, and $s'3$, do not need to be re-identified in the new implementation. In order to re-identify the added state $s'4$, the re-test sequence $r.\alpha4.W4 = r.bc.bb$ is selected using Formula (3-1a). If the modified implementation passes this sequence then there is a one-to-one mapping between states of the modified specification and implementation that are bb-equivalent.

Afterwards, in order to test the modified transition ($s'2$-c/f->$s'4$) whose head state $s'2$ is in S_u, the sequence $r.\alpha2.c.W4 = r.b.c.bb$ is selected using Formula (1-a). Moreover, the following re-testing test sequences are selected using Formula (3-1c) in order to check the outgoing transitions from state $s'4 \in S \backslash S_u$: $r.\alpha4.a.W4+r.\alpha4.b.W4+r.\alpha4.c.W3 = r.bc.a.bb+r.bc.b.bb+r.bc.c.bb$

Consequently, the re-testing test suite has sequences of total length 18. The traditional HIS method derives a test suite of length 32 if the whole specification of M_S' is considered for test derivation.

5. FURTHER RESEARCH WORK

We have extended the re-testing methods presented in this paper for the case when the system implementation may have more states than its specification. Moreover, we are adapting these methods for a system modeled as an Extended Finite State Machine (EFSM). The problem here is to find an appropriate way for re-testing both the control flow and the data flow parts of a modified EFSM. Finally, we are investigating how the re-testing methods can be applied to a labeled transition system (LTS).

REFERENCES

[1] T. S. Chow, 'Test design modeled by finite-state machines'. *IEEE Trans. SE*-4, No.3, pp. 178-187, 1978.

[2] K. El-Fakih, N, Yevtushenko, and G. v. Bochmann, 'Re-testing Based on Finite State Model', Technical Report, University of Ottawa, 2001.

[3] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, 'Test selection based on finite state models', *IEEE Trans. SE*-17, No. 6, 591-603, 1991.

[4] A. Gill, *Introduction to the Theory of Finite-State Machines.* McGraw-Hill, 1962.

[5] I.Koufareva, A.Petrenko, N.Yevtushenko. Test generation driven by user-defined fault models. In *Proc. of the IFIP TC6 12th International Workshop on Testing of Communicating Systems*, Hungary, pp. 215-233, 1999.

[6] A. Petrenko, *Checking experiments with protocol machines*, Proceedings of the IFIP Fourth International Workshop on Protocol Test Systems, the Netherlands, 1991, pp. 83-94.

[7] A. Petrenko and N. Yevtushenko,'Test suite generation for a FSM with a given type of implementation errors', *Proc. of the 12th IWPSTV*, pp. 229-243, 1992.

[8] A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das, 'Nondeterministic state machines in protocol conformance testing', In *Proc. of the IFIP Sixth International Workshop on Protocol Test Systems*, France, pp. 363-378, 1993.

[9] K. Sabnani and A. Dahbura,'A protocol test generation procedure'. C*omputer Networks and ISDN Systems*, Vol. 15, No. 4, 285-297, 1988.

[10] S. Schach, *Software Engineering*, Boston:Aksen Assoc., 1992.

[11] M. Yannakakis and D. Lee, 'Testing finite state machines: fault detection', *Journal of Computer and System Sciences*, 50, 1995, pp. 209-227, 1995.