An Improved Conformance Testing Method

Rita Dorofeeva[1], Khaled El-Fakih[2], and Nina Yevtushenko[1]

[1] Tomsk State University, Russia
[2] American University of Sharja, UAE
drf@kitidis.tsu.ru, kelfakih@aus.ac.ae,
yevtushenko@elefot.tsu.ru

Abstract. In this paper, we present a novel conformance test suite derivation method. Similar to the HIS method, our method uses harmonized state identifiers for state identification and transition checking and can be applied to any reduced possibly partial deterministic or nondeterministic specification FSM. However, in contrast with the HIS method, in the proposed method appropriate state identifiers are selected on-the-fly (for transition checking) in order to shorten the length of the obtained test suite. Application examples and experimental results are provided. These results show that the proposed method generates shorter test suites than the HIS method. Particularly, on average, the ratio of the length of the test suites derived using the proposed method over the length of corresponding suites derived using the HIS method is 0.66 (0.55) when the number of states of an implementation equals to (is greater than) the number of states of the specification. These ratios are almost independent of the size of specifications.

1 Introduction

Many FSM-based test derivation methods have been developed for conformance testing of communication protocols and other reactive systems [2,3,10,12,14,15,17]. Well-known methods are called the W [2, 14], partial W (Wp) [3], HIS [10,17], and generalized Wp (GWp) [7, 8] test derivation methods. For related surveys the reader may refer to [1,6,13,16]. In [2,3,10,14,15,17] testing methods, one usually assumes that not only the specification, but also the implementation can be modeled as a deterministic FSM, while in [7,8] the specification and the implementation are modeled as non-deterministic FSMs (NFSMs). If the behavior of a (deterministic/non-deterministic) implementation FSM is different than the specified behavior, the implementation contains a fault.

The above methods, each provides the following fault coverage guarantee: If the specification can be modeled by a (reduced) FSM with n states and if a corresponding implementation can be modeled by an FSM with at most m states, where m is larger or equal to n, then a test suite can be derived by the method (for this given m) and the implementation passes this test suite if and only if it conforms (i.e. is equivalent) to the specification. A test suite is called *m-complete* [11] if it detects any nonconforming implementation with at most m states. Guessing the bound of m is an intuitive process based on the knowledge of a specification, the class of implementations which have to be tested for conformance and their interior structure [1]. All of the above methods assume that a reliable reset is available for each implementation under test (written as ‘*r*’). This implies that a test suite can be composed of several individual test cases, each starting with the reset operation.

The HIS, Wp, and UIOv methods are modifications of the so-called W method. All these methods have two phases. Tests derived for the first phase check that each state presented in the specification also exists in the implementation, while tests derived for the second phase check all (remaining) transitions of the implementation for correct output and ending state as defined by the specification. For identifying the state during the first phase and for checking the ending states of the transitions in the second phase, certain state distinguishing input sequences are used. The only difference between the above methods is how such distinguishing sequences are selected. In the original W method, a so-called characterization set *W* is used to distinguish the different states of the specification. The Wp method uses the *W* set during the state identification phase (the first phase) while only an appropriate subset, namely a corresponding state identifier, is used when checking the ending state of a transition. In the UIOv method, which is a proper sub-case of the Wp method, the ending state of a transition is identified by the output obtained in response to a single input sequence. Such a Unique Input/Output sequence, called *UIO*, allows distinguishing the expected ending state from all other states of the specification. However, a *UIO* sequence may not exist for some states of a given specification FSM. Moreover, a *W* set also may not exist for a partially specified specification [16,17]. In this case, only the HIS method can be used where a family of state identifiers [9,10,16] is used for state identification as well as for transition checking.

The GWp method is a generalization of the Wp method to the case when the system specification and implementation are modeled as non-deterministic FSMs. For nondeterministic FSM implementations, in order to guarantee a full-fault detection power, the GWp method assumes that all possible observations of the non-deterministic implementation to a given test can be obtained by repeatedly executing this test. This assumption is called the *complete testing assumption* [7,8]. The GWp method uses a characterization set *W* to distinguish the different states of the specification. However, a *W* set may not exist for partially specified NFSMs. In this case, only the generalized HIS method [8] can be used where a family of harmonized state identifiers is used instead of a characterization set for state identification and transition checking.

The length of a derived test suite essentially depends on how a family of state identifiers is selected. In the above methods, for every state of the specification FSM, only one state identifier is selected (in advance) for testing all the incoming transitions of the state. In this paper, we propose an improved method that for every incoming transition of a state selects (on-the-fly) an appropriate state identifier that shortens the length of the resulting test suite. Our method generalizes the method (called H method) originally proposed for complete deterministic FSMs [5]. First we extend the H method for partial deterministic machines and we present more detailed sufficient conditions for having a complete test suite when the system specification and implementation have equal number of states. Then, we experiment with the extended H (hereafter also called as H method) method in order to compare the length of its test suites with test suites derived using the HIS method. The experiments are conducted for the case when $m = n$ and for the case when $m > n$. Experiments with the case when $m = n$ show that on average, the ratio of the length of H over the length of the HIS test suites is 0.66 and experiments with the case when $m = n + 1$ and $m = n + 2$ show that on average this ratio is 0.55. Moreover, the experiments show that these ratios are almost independent of the size of the specification machines. Finally, we extend the H test derivation method for partial nondeterministic machines. We note that the H method, as the HIS, generates complete test suites and is applicable to any complete or partial reduced specification machine.

This paper is organized as follows. Section 2 defines notations for describing finite state machines. Section 3 includes the H method for deterministic partial FSMs and Section 4 includes related experimental results. Section 5 includes the generalization of the H method for non-deterministic partial machines and Section 6 concludes the paper.

2 Finite State Machines

A non-deterministic finite state machine (NFSM) is an initialized non-deterministic Mealy machine that can be formally defined as follows. A *non-deterministic finite state machine M* is a 6-tuple $\langle S, X, Y, h, D_M, s_0\rangle$, where S is a finite nonempty set of states with s_0 as the initial state; X and Y are input and output alphabets; D_M is the specification domain that is a subset of $S\times X$; and $h\colon D_M \rightarrow 2^{S\times Y}\backslash\varnothing$ is a behavior function where $2^{S\times Y}$ is the set of all subsets of the set $S\times Y$. The behavior function defines the possible transitions of the machine. Given a present state s_i and an input symbol x, each pair $(s_j,y)\in h(s_i,x)$ represents a possible transition to the next state s_j with the output y.

An NFSM M is *observable* if for each pair $(s, x) \in D_M$ and each output y there is at most one state $s'\in S$ such that $(s', y) \in h(s, x)$. In this paper, we consider only observable NFSMs. Each NFSM is known to have an observable FSM with the same behavior [7, 8]. If $D_M = S\times X$ then M is said to be a *complete* FSM; otherwise, it is called a *partial* FSM. In the complete FSM we omit the specification domain D_M, i.e. complete FSM is 5-tuple $M = \langle S, X, Y, h, s_0\rangle$. If for each pair $(s, x) \in D_M$ it holds that $|h(s, x)| = 1$ then FSM M is said to be *deterministic*. In the deterministic FSM (DFSM) M instead of behavior function h we use two functions, transition function δ_M: $D_M \rightarrow S$ and output function λ_M: $D_M \rightarrow Y$.

We use 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 *starting* state of the transition, while s_j is said to be the *ending* state of the transition. If we are not interested in the output we write "s_i-x->s_j" when an input x is applied at state s_i.

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 including the empty word ε while Z^m denotes the set of all the words of length m. Let V be a set of words over alphabet Z. The prefix closure of V, written *Pref*(V), consists of all the prefixes of all words in V, i.e. $Pref(V) = \{\alpha \mid \exists\ \gamma\ (\alpha.\gamma \in V)\}$. The set V is *prefix-closed* if $Pref(V) = V$.

As usual, the behavior function h of a FSM M can be extended to the set X^* of finite input sequences. Given state s and input sequence $x_1\ldots x_k$, the pair $(s', y_1\ldots y_k) \in$

$h(s, x_1 \ldots x_k)$ if and only if there exists states $s_1', \ldots, s_{k+1}'$ such that $s_1' = s$ and $(s_{j+1}', y_j) \in h(s_j', x_j)$ for each $j = 1, \ldots, k$. In this case, the sequence $x_1 \ldots x_k$ is called a *defined input sequence* at state s. The set of all defined input sequences at state s of M is denoted $\mathbf{DIS}_M(s)$; while the set of defined input sequences at the initial state is denoted $\mathbf{DIS}_M$, for short.

Given an input sequence $\alpha = x_1 \ldots x_k$ and an FSM M, we let the set $out_M(s, \alpha)$ denote the set of output projections (i.e. responses) of M to the input α. Formally, $out_M(s, \alpha) = \{\gamma | \exists s' \in S [(s', \gamma) \in h(s, \alpha)]\}$. If M is deterministic then $|out_M(s, \alpha)| \leq 1$. The FSM is called *connected* if for each state $s \in S$ there exists an input sequence α_s that takes the FSM from the initial state to state s. The sequence α_s is called a *transfer* sequence for the state s. We further consider only *connected* FSMs. A set Q of input sequences is called a *state cover set* of FSM M if for each state s_i of S, there is an input sequence $\alpha_i \in Q$ such that s_1-α_i->s_i. A state cover set exists for every connected FSM. We further consider prefix-closed state cover sets, i.e., we include the empty sequence ε in Q.

Let $M = (S, X, Y, h_M, D_M, s_1)$ and $I = (T, X, Y, h_I, D_I, t_1)$ be two FSMs. In the following sections M usually represents a specification while I denotes an implementation. We say that state t of I is *quasi-equivalent* to state s of M [4, 10, 8], written $t \sqsupseteq s$, if $\mathbf{DIS}_M(s) \subseteq \mathbf{DIS}_I(t)$, and for each input sequence $\alpha \in \mathbf{DIS}_M(s)$ it holds that $out_M(s, \alpha) = out_I(t, \alpha)$. In other words, FSM I at state t can have "more defined" behavior than FSM M at state s. However, for each defined input sequence at state s, the output responses of FSMs M and I coincide. FSM I is *quasi-equivalent* to M if $t_1 \sqsupseteq s_1$. We also say that states s and t are *distinguishable*, written $s \not\cong t$, if there exists an input sequence $\alpha \in \mathbf{DIS}_M(s) \cap \mathbf{DIS}_I(t)$ such that $out_M(s, \alpha) \neq out_I(t, \alpha)$; the sequence α is said to *distinguish* the states s_j and t_i. Two FSMs M and I are said to be *distinguishable* if their initial states are distinguishable. An FSM is said to be *reduced* if its states are pair-wise distinguishable.

When testing NFSMs, the specification M of the given system is assumed to be a partial/complete non-deterministic finite state machine while an implementation I of M is assumed to be a complete and non-deterministic. However, it is assumed that the specification M has so-called *harmonized traces* [10] such that for each input sequence $\alpha = x_1 \ldots x_k$ defined at the initial state and each two pairs $(s', y_1 \ldots y_k)$, $(s'', y_1 \ldots y_k) \in h_M(s_1, x_1 \ldots x_k)$ the sets of defined input sequences at states s' and s'' coincide. The reason is a test suite is derived in advance and each input sequence is applied independently of the output response of an implementation at hand. When testing DFSMs, the specification M of the given system is assumed to be a deterministic partial/complete finite state machine while an implementation I of M is assumed to be complete and deterministic.

We say that implementation I *conforms to* the specification M if and only if FSM I is quasi-equivalent to M. In other words, for each input sequence that is defined in the specification the output responses of M and I coincide [4,16,8]. Otherwise, I is called a *nonconforming* (or *faulty*) implementation of M. In this case, an input sequence α that distinguishes initial states of FSMs I and M is said to *distinguish* the implementation I from the specification M or α is said to *detect* the faulty implementation I.

Given the input alphabet X, we denote $J_m(X)$ the set of all complete deterministic machines over the input alphabet X with up to m states. Given a deterministic specification FSM M, a *test suite TS* is a finite set of finite input sequences of the FSM M. A test suite TS is *m-complete* for the specification FSM M if for each implementation $I \in J_m(X)$ that is distinguishable from M, there exists a sequence in TS that distinguishes M and I.

3 An Improved Test Generation Method

Given a deterministic reduced possibly partial specification FSM M with n states, in this section, we first establish sufficient conditions for having an m-complete test suite. This is done for the cases when $m > n$ and when $m = n$. Based on these conditions, in the following two subsections we present a novel test derivation method with related experimental results. In Section 5, we generalize the method for the case when the specification and implementation machines are non-deterministic.

3.1 Sufficient Conditions for an M-Complete Test Suite

Given a reduced deterministic specification machine M with n states, the following theorem establishes sufficient conditions for a given test suite to be m-complete assuming that $m \geq n$. The theorem extends a related theorem given in [5] for partial deterministic specification machines.

Theorem 1. Given a reduced deterministic specification M with n states and a state cover set Q of M, let TS be a finite set of defined finite input sequences of M that contains the set of sequences $Q.X^{m-n+1} \cap \mathbf{DIS}_M$. The test suite TS is m-complete if the following conditions hold:

1. For each two (different) states of M reachable through sequences α and β in Q, TS has sequences $\alpha.\gamma$ and $\beta.\gamma$ where γ is a distinguishing sequence of the states $\delta_M(s_1, \alpha)$ and $\delta_M(s_1, \beta)$ reachable by the sequences α and β, respectively.
2. For each sequence $\alpha.\beta$, $\alpha \in Q$, $|\beta| = m - n + 1$, and each non-empty prefix β_1 of β that takes the specification FSM M to state s from state $\delta_M(s_1, \alpha)$, TS has the sequences $\alpha.\beta_1.\gamma$ and $\omega.\gamma$, where $\omega \in Q$ and $\delta_M(s_1, \omega) \neq s$, and γ is a distinguishing sequence of states $\delta_M(s_1, \alpha\beta_1)$ and $\delta_M(s_1, \omega)$.
3. For each sequence $\alpha.\beta$, $\alpha \in Q$, $|\beta| = m - n + 1$, and each two non-empty prefixes β_1 and β_2 of β that take the specification FSM M from state $\delta_M(s_1, \alpha)$ to two different states, TS has sequences $\alpha.\beta_1.\gamma$ and $\alpha.\beta_2.\gamma$, where γ is a distinguishing sequence of states $\delta_M(s_1, \alpha\beta_1)$ and $\delta_M(s_1, \alpha\beta_2)$.

Proof. Consider a test suite TS that satisfies the conditions of the theorem and assume that there exists a complete FSM $I = \langle T, X, Y, \delta_I, \lambda_I, t_1\rangle$ with m states that is distinguishable from the specification FSM $M = \langle S, X, Y, \delta_M, \lambda_M, D_M, s_1\rangle$ but for each input sequence of the set TS, the output responses of M and I to the input sequence coincide. Let P be the set of states that are reachable in I via sequences of the state cover set Q, and ν be a shortest input sequence from some state $\delta_I(t_1, \alpha_j)$ of the set P that distinguishes states $\delta_M(s_1, \alpha_j)$ and $\delta_I(t_1, \alpha_j)$. By definition, TS has a sequence $\alpha_j\beta$, where β has length $m - n + 1$ and β is a prefix of ν.

Consider the set R of sequences that is the union of sequences in the state cover set and the set of sequences $\alpha_j\beta'$ over all non-empty prefixes β' of β. The number of such sequences equals to $n + (m - n + 1) = m + 1$ and since I has at most m states there are two sequences in the set that take FSM I from the initial state to the same state. Let $R = \{\alpha_1, \ldots, \alpha_n, \ldots, \alpha_{m+1}\}$ where $\alpha_i = \alpha_j\beta_i$ for $i = n + 1, \ldots, m + 1$, and $\delta_I(t_1, \alpha_i) = \delta_I(t_1, \alpha_r)$.

Considering i and r, there are three possible cases: $i, r \le n$; $i \le n < r$; or $i, r > n$.

1) $i, r \le n$. In this case, $\alpha_i, \alpha_r \in Q$ and the set TS has sequences $\alpha_i\gamma$ and $\alpha_r\gamma$ where γ distinguishes states $\delta_M(s_1, \alpha_i)$ and $\delta_M(s_1, \alpha_r)$. Thus, this case is not possible for FSM I that passes the test TS.

2) $i \le n < r$. In this case, $\alpha_i \in Q$ and the trace ν', where ν' is obtained from ν by deleting the prefix β_r, distinguishes states $\delta_M(s_1, \alpha_i)$ and $\delta_I(t_1, \alpha_r)$. The latter contradicts the fact that the trace ν that contains β as a prefix is a shortest trace with such feature.

3) $i, r > n$. In this case, the trace ν could be shortened by deleting the part between two states $\delta_M(s_1, \alpha_i)$ and $\delta_M(s_1, \alpha_r)$.

Thus, given an FSM I with at most m states that is not quasi-equivalent to M, there exists an input sequence of the test TS such that output response of I to the sequence is different from that of the specification FSM M, i.e., TS is m-complete. □

According to Theorem 1, given a state s of M, different state identification sequences can be used when checking different incoming transitions of state s. When $m = n$, Theorem 1 can be refined and the following theorem establishes sufficient conditions for a given test suite to be n-complete.

Theorem 2. Given a reduced deterministic specification M with n states and state cover set Q of M, let TS be a finite set of defined finite input sequences of M that contains the set $Q.X \cap \mathbf{DIS}_M$. The set TS is n-complete if the following conditions hold:

1. For each two different states of M reachable through sequences α and β in Q, TS has the sequences $\alpha.\gamma$ and $\beta.\gamma$ where γ is a distinguishing sequence of the states $\delta_M(s_1, \alpha)$ and $\delta_M(s_1, \beta)$.
2. For each defined transition (s, x) of the specification M, TS has a sequence $\alpha.x$ with the following properties:
   a) $\delta_M(s_1, \alpha) = s$.
   b) For each state reachable through a sequence $\beta \in Q$ such that state $\delta_M(s_1, \beta) \ne s$, TS has sequences $\alpha.\gamma$ and $\beta.\gamma$ where γ is a distinguishing sequence of states s and $\delta_M(s_1, \beta)$.
   c) For each state reachable through sequence $\beta \in Q$ such that state $\delta_M(s_1, \beta) \ne \delta_M(s, x)$, TS has sequences $\alpha.x.\gamma$ and $\beta.\gamma$, where γ is a distinguishing sequence of states $\delta_M(s, x)$ and $\delta_M(s_1, \beta)$.

In the following section, we consider a simple application example that shows how test suites derived by other methods can be shortened by use of Theorems 1 and 2. We also illustrate by an example that the conditions stated in Theorems 1 and 2 are not necessary conditions.

3.2 Application Example

Consider the specification FSM shown in Fig. 1. We derive a 4-complete test suite based on the HIS method [10, 17] using $Q = \{\varepsilon, a, b, c\}$ as a state cover set and $F = \{H_1, H_2, H_3, H_4\}$ with $H_1 = \{a, bb\}$, $H_2 = \{a, b\}$, $H_3 = \{a\}$, $H_4 = \{a, bb\}$, as a separating family of state identifiers. For state identification, we use the sequences: $r.\varepsilon.H_1 + r.a.H_3 + r.b.H_4 + r.c.H_2$. For testing transitions we use the sequences: $r.a.H_3 + r.b.H_4 + r.c.H_2 + r.a.a.H_2 + r.a.b.H_1 + r.b.a.H_3 + r.b.b.H_2 + r.b.c.H_3 + r.c.a.H_4 + r.c.b.H_3$. We replace the H's in the above sequences by their corresponding values and then remove from the obtained set those sequences that are proper prefixes of other sequences. The obtained 4-complete test suite $TS_{HIS} = \{raaa, raab, raba, rabbb, rbaa, rbba, rbbb, rbca, rcaa, rcabb, rcba\}$ with total length 46. However, due to Theorem 1, we do not need to append sequence *r.aa* with *a*, as *b* already distinguishes state 2 = $\delta_M(1, aa)$ from any other state reachable through the sequences of the state cover set. For the same reason, without loss of the completeness of the test suite the following sequences can be deleted from TS_{HIS}: *rbba* and *rcaa*. Moreover, transition 4–*b*-> 2 can be checked by the sequences *rcabb*, while transition 2 – *b* –> 3 can be checked by the sequence *raaba*; thus, the sequence *rcba* and *rbbb* can be deleted from the test suite. As a result, we obtain a 4-complete test suite {*raaba*, *raba*, *rabbb*, *rbaa*, *rbca*, *rcabb*} with total length 27.

We further show that the conditions of Theorems 1 and 2 are not necessary conditions. The reason is that two states can be implicitly distinguished if their successor states under some input are different. Consider the FSM *B* shown in Fig. 2.

	1	**2**	**3**	**4**
a	3/1	4/1	2/0	3/1
b	4/0	3/1	1/0	2/0
c	2/1	-	-	3/1

Fig. 1. Specification FSM *M*

	1	**2**	**3**
a	2/0	1/1	2/1
b	3/0	1/1	3/1

Fig. 2. Specification FSM *B*

The FSM B has the set $Q = \{\varepsilon, a, b\}$ as a state cover. We consider the test suite TS_1 of all prefixes of the set $\{raaa, rabb, rbaba, rbbab\}$; the set TS_1 contains all the sequences of the set $r.Q.\{a, b\}$. We first observe that states 2 and 3 that are reachable through sequences a and b in Q are not distinguished with suffixes *aa*, *bb* and *aba*, *bab* applied after *ra* and *rb* in the test suite. Nevertheless, if an implementation at hand passes the test suite TS_1 then the states reachable after the sequences *ra* and *rb* are different. Otherwise, the states reachable after the sequence *raa* (*rab*) and after the sequence *rba* (*rbb*) coincide and thus we have four different output responses to the sequences *r.a* and *r.b*:

	a	***b***
t_1	0	0
$\delta_B(t_1, a) = \delta_B(t_1, b)$	1	1
$\delta_B(t_1, a.a) = \delta_B(t_1, b.a)$	0	1
$\delta_B(t_1, a.b) = \delta_B(t_1, b.b)$	1	0

If an implementation I has at most three states and passes TS_1, we can draw the following conclusions:

a) States $t_2 = \delta_I(t_1, a)$ and $t_3 = \delta_I(t_1, b)$ of I are two different states;
b) Input a distinguishes the initial state of I from the other states of I;
c) $\delta_I(t_2, a) = \delta_I(t_2, b) = t_1$.

Thus, $\delta_I(t_3, a) = t_2$ since $\lambda_I(t_3, ba) = 11$, and $\delta_I(t_3, b) = t_3$ since $\lambda_I(t_2, ab) = 10$ and $\delta_I(t_2, a) = t_1$. Therefore, any implementation that passes TS_1 is equivalent to the given specification FSM B, i.e., TS_1 is a 3-complete test suite.

As demonstrated by the above example, the possibility to distinguish two states based on their successor states depends on the number of states of an implementation at hand. Based on this, more rigorous analysis is needed to determine related conditions that can be used for shortening the length of a test suite.

3.3 Test Derivation Method

Let A be a specification FSM, $A = (S,X,Y,\delta,\lambda,s_0)$, where $|S|=n$. Below we present a test generation method that derives an *m-complete* test suite for A, where $m \geq n$.

Algorithm 1. Test Generation Method

Input : A reduced deterministic specification FSM $M = (S,X,Y,\delta,\lambda,D_M,s_0)$ with n states, a prefix-closed state cover set Q of M, and an upper bound m on the number of states of an implementation FSM of M, where $m \geq n$.

Output : An m-complete test suite TS

Step 1. Derive the set of sequences $TS = QPref(X^{m-n+1}) \cap \mathbf{DIS}_M$

Step 2. For each two sequences α_i and α_j of the state cover set Q check if the set TS has sequences $\alpha_i.\omega$ and $\alpha_j.\omega$ such that ω distinguishes states $\delta(s_0, \alpha_i)$ and $\delta(s_0, \alpha_j)$ in the specification FSM. If there are no such sequences select a sequence ω that distinguishes states $\delta(s_0, \alpha_i)$ and $\delta(s_0, \alpha_j)$ and add to TS sequences $\alpha_i.\omega$ and $\alpha_j.\omega$.

Step 3. For each sequence $\alpha_i.\beta \in QPref(X^{\leq m-n+1}) \cap \mathbf{DIS}_M$, $\alpha_i \in Q$, let s_i be the state reachable from the initial state of M by the sequence α_i. Check if the set TS has sequences $\alpha_i.\beta.\omega$ and $\alpha_j.\omega$, $\alpha_j \in Q$, $\delta(s_0, \alpha_j) \neq s$, such that ω distinguishes states $\delta(s_0, \alpha_i.\beta)$ and $\delta(s_0, \alpha_j)$ in the specification FSM. If there are no such sequences select sequence ω that distinguishes states $\delta(s_0, \alpha_i.\beta)$ and $\delta(s_0, \alpha_j)$ and add to TS sequences $\alpha_i.\beta.\omega$ and $\alpha_j.\omega$.

Step 4. For each sequence $\alpha_i.\beta \in QPref(X^{\leq m-n+1}) \cap \mathbf{DIS}_M$, $\alpha_i \in Q$, and each two non-empty prefixes β_1 and β_2 of the sequence β, check if the set TS has sequences $\alpha_i.\beta_1.\omega$ and $\alpha_i.\beta_2.\omega$ such that ω distinguishes states $\delta(s_0, \alpha_i.\beta_1)$ and $\delta(s_0, \alpha_i.\beta_2)$ in the specification FSM. If there are no such sequences select sequence ω that distinguishes states $\delta(s_0, \alpha_i.\beta_1)$ and $\delta(s_0, \alpha_i.\beta_2)$ and add to TS sequences $\alpha_i.\beta_1.\omega$ and $\alpha_i.\beta_2.\omega$.

Due to Theorem 2, the following statement holds.

Theorem 3. The set of input sequences TS obtained with Algorithm 1 is an m-complete test suite for the given specification FSM M.

Intuitively, the above method uses an appropriate sequence for testing each transition of the implementation. The prefix of the sequence takes the implementation to the starting state of the tested transition and its suffix distinguishes the expected final state of the tested transition from all other states of the specification using an appropriate state identification sequence. Unlike the HIS method, the identification sequence of the starting state of the tested transition has to be harmonized only with other state identification sequences of the states of the state cover set Q and of the states reachable from the initial state by the prefix of the sequence. Moreover, for the same state of the specification, we can use different state identification sequences when testing different transitions.

To illustrate the method we derive a 5-complete test suite for the specification FSM in Figure 1. We first use the HIS method to obtain an 5-complete test suite and obtain the set {*raaaa*, *raaabb*, *raaba*, *rabaa*, *rabba*, *rabbbb*, *rabca*, *rabcb*, *rbaaa*, *rbaab*, *rbaba*, *rbabbb*, *rbbaa*, *rbbabb*, *rbbba*, *rbcaa*, *rbcab*, *rbcba*, *rbcbbb*, *rcaaa*, *rcaba*, *rcabb*, *rcaca*, *rcbaa*, *rcbab*, *rcbba*, *rcbbbb*} with total length 141. We use state identifiers {*a*} and {*b*} to check states 3 and 2, as both of them are applied after sequences of the state cover set Q, state identifier *bb* to check state 4, state identifiers {*a*, *bb*} to check state 1, and obtain the test suite {*raaabb*, *raaba*, *rabaa*, *rabbbb*, *rabcb*, *rbaab*, *rbaba*, *rbabbb*, *rbbabb*, *rbbba*, *rbcab, rbcba*, *rbcbbb*, *rcaaa*, *rcabb*, *rcaca*, *rcbab*, *rcbba*, *rcbbbb*} that is 5-complete and has total length 101.

4 Experimental Results

In this section we experiment with the HIS and H methods in order to compare the length of their test suites. Table 1 provides a comparison between the length of the test suites obtained by these methods for the case when the number of states of an implementation of a given system equals to the number of states of the given specification (i.e. $m = n$) and Table 2 provides a comparison for the case when $m > n$. The comparison in Tables 1 and 2 is based on randomly generated completely specified reduced deterministic specifications with a varying number of states (n).

Each row of Tables 1 (2) corresponds to a group of 50 randomly generated completely specified reduced specifications. For each of these specifications we use the HIS and H methods to derive corresponding test suites. Then, we calculate the average length of the test suites generated for each group using each of these methods as shown in Columns IV and V (V and VI), respectively.

Table 1. A summary of experiments for the case when number of inputs/outputs equals to 10 and *m*=*n*

I- Groups of 50 Experiments	II- Number of States	III- Number of Transitions	IV- Average Length HIS Test Suites	V- Average Length H Test Suites
1	30	300	2243	1568
2	50	500	4375	2852
3	60	600	5490	3656
4	70	700	6694	4344
5	80	800	7892	5216
6	90	900	9194	6078
7	100	1,000	10503	6880

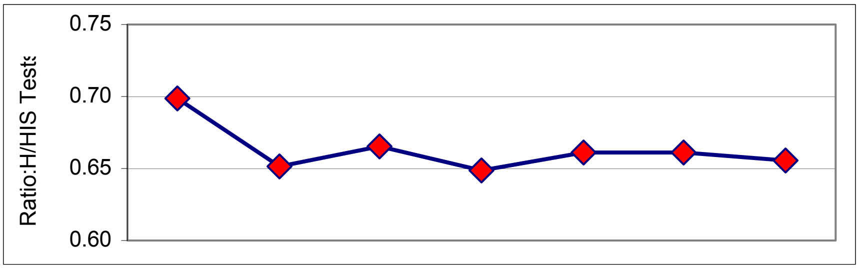

Fig. 3. Rations of average length H/HIS Test Suites when $m = n$ for the experiments in Table 1

Figure 3 depicts the ratios of length of the test suites of the H method over the length of the HIS based test suites for the experiments shown in Table 1. On average, the H test suites are 0.66 percent of the HIS test suites. Moreover, according to the experiments this ratio is almost independent of the size of the specification.

Table 2. A summary of experiments for the case when number of inputs/outputs equals 4 and *m*=*n*+1 and *m*=*n*+2

I- Groups of 50 Experiments	II- Number of States of the specification	III- Number of States of an implementation	IV- Number of Transitions	V- Average Length HIS Test Suites	VI- Average Length H Test Suites
1	10	11	44	1399	883
2	20	21	84	3773	2212
3	30	31	124	6432	3560
4	40	41	164	9272	5188
5	50	51	204	12494	6837
6	10	12	48	6790	4181
7	20	22	88	17238	9565
8	30	32	128	29860	15115
9	40	42	168	44137	21919
10	50	52	208	58949	28813

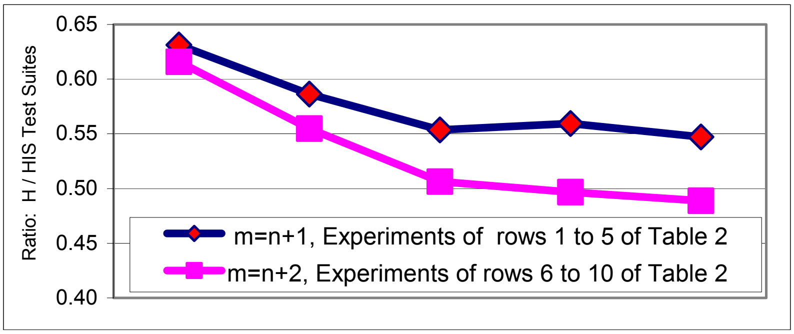

Fig. 4. Ratios of average length H/HIS Test Suites when *m*=*n*+1 and *m*=*n*+2

Figure 4 depicts the ratios of length of the test suites of the H method over the length of the HIS based test suites for the experiments depicted in rows 1 to 5 and rows 6 to 10 of Table 2. On average, the H test suites are 0.55 percent of the HIS test suites. Moreover, according to the experiments this ratio slightly decreases as the size of the specification increases.

5 Generalizing the H Method for Nondeterministic Machines

When the specification FSM M is reduced, possibly partial, nondeterministic with harmonized traces, the fault domain $R_m(X)$ contains all complete observable NFSM implementations of M, defined over the input alphabet X of M, with at most m states. A test suite TS is *m-complete* for M if for each implementation $I \in R_m(X)$ that is distinguishable from M, there exists a sequence in TS that distinguishes M and I.

Due to the complete testing assumption [7, 8], the procedure used for deriving an m-complete test suite for deterministic machines can be adapted for deriving an m-complete test suite for nondeterministic machines. First, we derive a state cover set of the NFSM specification $M = (S, X, Y, h_M, D_M, s_1)$ [7, 8]. However, in this case, differently from deterministic FSMs, the number of sequences in a state cover set can be less than the number of states n of the specification machine. This is due to the fact that the specification machine in response to a single defined input sequence, repeatedly applied at the initial state, can reach several states and produce in response different output sequences.

For each state $s_i \in S$, we consider the sequences $\alpha_i.X^{m-n+1} \cap \mathbf{DIS}_M(s_i)$, where $\alpha_i \in Q$ is the sequence of Q that takes the NFSM specification to state s_i from the initial state and $\mathbf{DIS}_M(s_i)$ is the set of all defined input sequences at state s_i. We let $Q.X^{m-n+1} \cap \mathbf{DIS}_M$ denote the set of all obtained sequences. Since the specification FSM has harmonized traces, the sets of defined input sequences of the states reachable by any initially defined input sequence coincide. For every sequence $\alpha_i \in Q$, we denote $S_i \in S$ the subset of states for which we use $\alpha_i \in Q$. The subsets S_i form a partition of the set S.

As an application example of the HIS method [8], consider the NFSM M shown in Fig. 5. M admits the set $\{\varepsilon, a, b\}$ as a state cover set. The NFSM M is reduced; state 3 can be distinguished from all other states by the input sequence a, state 2 can be distinguished from all other states by the input sequence aa, and states 1 and 4 can be distinguished by the input sequence b. Moreover, M has harmonized traces since it has the same set of defined inputs at states 2 and 3. Let $m = 4$. Then the set $Q.X^{m-n+1} \cap \mathbf{DIS}_M = \{a, b, c, aa, ac, ba, bb, bc\}$.

The FSM M has the following sets of harmonized state identifiers, $H_1 = \{aa, b\}$, $H_2 = \{aa\}$, $H_3 = \{a\}$ and $H_4 = \{aa, b\}$. For each sequence in the set $Q.X^{m-n+1} \cap \mathbf{DIS}_M$, we determine the states reached by this sequence and append the sequence with corresponding sets of harmonized state identifiers. The union of all obtained test sequences is the 4-complete test suite $TS_{HIS} = \{raaaa, racaa, racb, rbaaa, rbbaa, rbbb, rbcaa, rcaa, rcb\}$ which is of length 40. Here we note that state 1 has also the sequence cc as a state identifier which is shorter than the sequences of H_1, but cc is not used since it is not harmonized with the identifiers of all other states. We note that the GWp [7] method can not be applied to this example since M does not have a characterization set. States 1 and 4 of M can be distinguished only by an input sequence *b* or by an input sequence with the head symbol *c*. However, by direct inspection, one can observe that states 2 and 3 can not be distinguished with these sequences.

	1	**2**	**3**	**4**
a	3/1; 2/0	2/1; 3/0	2/0	3/1; 2/0
b	4/0,1	-	-	1/1
c	1/1	1/0	1/0	2/1

Fig. 5. Partial NFSM specification *M*

Similar to Theorem 1, the following theorem allows us to use non-harmonized state identifiers when deriving an *m*-complete test suite.

Theorem 4. Given a set *TS* of defined input sequences of the specification NFSM *M*, let *TS* contain the set $Q.X^{m-n+1} \cap \mathbf{DIS}_M$. The set *TS* is *m*-complete if

- For each two sequences α_i and $\alpha_j \in Q$, for each two states $s_i \in S_i$ and $s_j \in S_i$, the set *TS* has sequences $\alpha_i\sigma$ and $\alpha_j\sigma$ where σ distinguishes states s_i and s_j;
- For each sequence $\alpha_i.\nu \in Q.(X^{\leq m-n+1}) \cap \mathbf{DIS}_M$, $\alpha_i \in Q$, each state $s_i \in S_i$, and each state *s* reachable from state s_i via the sequence ν and each state $s_j \in S_j$, $s_j \neq s$ and $\alpha_j \in Q$ the set *TS* has sequences $\alpha_i.\nu\sigma$ and $\alpha_j.\sigma$, where σ distinguishes state *s* from state s_j.
- For each sequence $\alpha_i.\beta$, $\alpha_i \in Q$, $|\beta| = m - n + 1$, each state $s_i \in S_i$ and each two non-empty prefixes β_1 and β_2 of β that take the specification FSM *M* from state s_i to the subsets of states P_1 and P_2, for each two states $s_1 \in P_1$ and $s_2 \in P_2$, *TS* has sequences $\alpha.\beta_1.\gamma$ and $\alpha.\beta_2.\gamma$, where γ is a distinguishing sequence of states s_1 and s_2.

In other words, similar to the case of deterministic FSMs, we do not need to derive in advance state distinguishing sequences for the specification FSM. These sequences can be derived on-the-fly starting with the set $Q.X^{m-n+1} \cap \mathbf{DIS}_M$. Moreover, different state identifiers can be used for checking incoming transitions of states. These identifiers do not have to be harmonized with the identifiers of all other states. In our working example, we observe that the state identifier {*cc*} can be used for identifying state 1 despite of the fact that this identifier is not harmonized with the identifiers of all other states. Due to the above theorem, the test suite TS_2 = {*raaaa*, *raccc*, *rbaaa*, *rbbcc*, *rbcaa*, *rbcc*, *rccc*} of length 33 is also 4-complete.

Algorithm 2. Test Generation Method

Input : The reduced nondeterministic specification FSM $M = (S,X,Y,\delta,\lambda,D_M,s_0)$ with *n* states and harmonized traces, a prefix-closed state cover set *Q* of *M*, and an upper bound *m* on the number of states of an implementation FSM, where $m \geq n$.

Output : An m-complete test suite TS

Step 1. Derive the set of sequences $TS = QPref(X^{m-n+1}) \cap \mathbf{DIS}_M$ and fix for each $\alpha_i \in Q$ the subset S_i of states for which we use the sequence α_i.

Step 2. For each two sequences α_i and α_j of the state cover set Q and each two states $s_i \in S_i$ and $s_j \in S_j$, check if the set TS has sequences $\alpha_i.\omega$ and $\alpha_j.\omega$ such that ω distinguishes states s_i and s_j in the specification FSM. If there are no such sequences select a sequence ω that distinguishes states s_i and s_j and add into TS sequences $\alpha_i.\omega$ and $\alpha_j.\omega$.

Step 3. For each sequence $\alpha_i.\beta \in QPref(X^{\leq m-n+1}) \cap \mathbf{DIS}_M$, $\alpha_i \in Q$, each state $s_i \in S_i$, each state s reachable from s_i via sequence β and each state $s_j \in S_j$, $\alpha_j \in Q$, check if the set TS has sequences $\alpha_i.\beta.\omega$ and $\alpha_j.\omega$, $s_j \neq s$, such that ω distinguishes states s_j and s in the specification FSM. If there are no such sequences select sequence ω that distinguishes states s_j and s and add to TS sequences $\alpha_i.\beta.\omega$ and $\alpha_j.\omega$.

Step 4. For each sequence $\alpha_i.\beta \in QPref(X^{\leq m-n+1}) \cap \mathbf{DIS}_M$, $\alpha_i \in Q$, each state $s_i \in S_i$, and each two non-empty prefixes β_1 and β_2 of the sequence β, let P_1 and P_2 be the sets of states reachable from state s_i via sequences β_1 and β_2. For each two states $s_1 \in P_1$ and $s_2 \in P_2$, check if the set TS has sequences $\alpha_i.\beta_1.\omega$ and $\alpha_i.\beta_2.\omega$ such that ω distinguishes states s_1 and s_2 in the specification FSM. If there are no such sequences select sequence ω that distinguishes states s_1 and s_2 and add to TS sequences $\alpha_i.\beta_1.\omega$ and $\alpha_i.\beta_2.\omega$.

Due to Theorem 4, the following statement holds.

Theorem 5. The set of input sequences TS obtained using Algorithm 2 is an m-complete test suite for the given specification FSM M.

6 Conclusion

An improved HIS based test derivation method has been presented in this paper. The method can be applied for any reduced possibly partial and nondeterministic specification machine. In comparison with the HIS method, in the proposed method state identifiers are derived on-the-fly and different state identifiers can be used when checking different incoming transitions of a state. Experimental results show that the proposed method returns shorter test suites than the HIS method. In particular, on average, the length of a test suite derived using the H method is 0.66% (0.55%) of the length of a test suite derived using the HIS method when the number of states of an implementation equals to (is greater than) the number of states of the specification. The length of a test suite returned by the proposed method essentially depends on the order in which transitions are checked. Accordingly, currently, we are incorporating into our method an optimization procedure that determines an order that provides a shortest length test suite.

References

1. G. v. Bochmann, A. Petrenko, “Protocol testing: review of methods and relevance for software testing,” *Proc. International Symposium on Software Testing and Analysis*, Seattle, 1994, pp. 109-123.
2. T. S. Chow, “Test design modeled by finite-state machines,” *IEEE Trans. SE*, vol. 4, no.3, 1978, pp. 178-187.
3. S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, “Test selection based on finite state models,” *IEEE Trans. SE*, vol. 17, no. 6, 1991, pp. 591-603.
4. AGill, *Introduction to the Theory of Finite-State Machines*, McGraw-Hill, 1962.
5. I. Koufareva, M. Dorofeeva. *A novel modification of W-method.* Joint Bulletin of the Novosibirsk computing center and A.P. Ershov institute of informatics systems. Series: Computing science, issue: 18, 2002, NCC Publisher, Novosibirsk. - PP. 69-81.
6. D. Lee and M. Yannakakis, “Principles and methods of testing finite state machines-a survey”, *Proceedings of the IEEE*, vol. 84, no. 8, 1996, pp. 1090-1123.
7. G. L. Luo, G. v. Bochmann, and A. Petrenko, “Test Selection Based on Communicating Nondeterministic Finite-State Machines Using a Generalized Wp-method”, *IEEE Transactions on Software Engineering*, 20(2):149–161, 1994.
8. G. Luo, A. Petrenko, G. v. Bochmann, “Selecting Test Sequences for Partially Specified Nondeterministic Finite State Machines”, *Proc. 7th IWPTS*, Japan, 1994.
9. A. Petrenko, “Checking experiments with protocol machines,” *Proc. 4th Int. Workshop on Protocol Test Systems*, 1991, pp. 83-94.
10. A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das, “Nondeterministic state machines in protocol conformance testing,” *Proc. of the IFIP 6th IWPTS*, France, 1993, pp. 363-378.
11. A. Petrenko and N. Yevtushenko, “On Test Derivation from Partial Specifications”, Proc. of the IFIP Joint International Conference, FORTE/PSTV'2000, Italy, 2000, pp. 85-102.
12. K. Sabnani and A. Dahbura, “A protocol test generation procedure,” Computer Networks and ISDN Systems, vol. 15, no. 4, 1988, pp. 285-297.
13. D. P. Sidhu, and T. K. Leung, “Formal methods for protocol testing: a detailed study,” *IEEE Trans. SE*, vol. 15, no. 4, 1989, pp. 413-426.
14. M. P. Vasilevskii, “Failure diagnosis of automata,” translated from Kibernetika, No.4, 1973, pp. 98-108.
15. S. T. Vuong, W.W.L. Chan, and M.R. Ito, “The UIOv-method for protocol test sequence generation,” *Proc. of the IFIP TC6 2nd IWPTS*, North-Holland, 1989, pp. 161-175.
16. M. Yannakakis and D. Lee, “Testing finite state machines: fault detection”, *Journal of Computer and System Sciences*, 50, 1995, pp. 209-227.
17. N. Yevtushenko and A. Petrenko, *Test derivation method for an arbitrary deterministic automaton*, Automatic Control and Computer Sciences, Allerton Press Inc., USA, #5, 1990.