Generating Reduced Tests for FSMs with Extra States

Adenilso Simão [1,2], Alexandre Petrenko [2] and Nina Yevtushenko [3]
[1] São Paulo University
São Carlos, São Paulo, Brazil
[2] Centre de recherche informatique de Montreal (CRIM)
Montreal, Quebec, Canada
[3] Tomsk State University
Tomsk, Russia

adenilso@icmc.usp.br, petrenko@crim.ca, ninayevtushenko@yahoo.com

Abstract. We address the problem of generating tests from a deterministic Finite State Machine to provide full fault coverage even if the faults may introduce extra states in the implementations. It is well-known that such tests should include the sequences in the so-called traversal set, which contains all sequences of length defined by the number of extra states. Therefore, the only apparent opportunity to produce shorter tests is to find within a test suite a suitable arrangement of the sequences in the inescapable traversal set. We observe that the direct concatenation of the traversal set to a given state cover, suggested by all existing generation methods with full fault coverage, results in extensive test branching, when a test has to be repeatedly executed to apply all the sequences of the traversal set. In this paper, we state conditions which allow distributing these sequences over several tests. We then utilize these conditions to elaborate a method, called SPY-method, which shortens tests by avoiding test branching as much as possible. We present the results of the experimental comparison of the proposed method with an existing method which indicate that the resulting save can be up to 40%.

1. Introduction

Finite State Machines (FSMs) have been used to model systems in many areas, such as hardware design, formal language recognition, conformance testing of protocols [1] and object-oriented software testing [2]. Regarding test generation, one of the main advantages of using FSMs is the existence of generation methods which guarantee full fault coverage: given a specification FSM with n states and any black-box implementation which can be modelled as an FSM with at most m states, $m \geq n$, the methods generate a test suite, often called m-complete test suite, which has the ability to detect all faults in any such implementations. In the particular case of $m = n$, there are many efficient methods which generate complete test suites [7] [3] [5] [10] [4].

However, on the other hand, in spite of the fact that the problem of generating *m*-complete test suites for $m > n$ is a longstanding one which can be traced back to the work of Moore [11] and Hennie [9], it has received much less attention compared to the problem of constructing *n*-complete test suites. One of the main reasons might be the fact that test generation becomes more challenging in the case of extra states. It is known that an *m*-complete test suite should include each sequence in the so-called traversal set, which contains all input sequences with $m - n + 1$ inputs [13]. Moreover, the traversal set should be applied to each state of the specification. Not surprisingly, all, not numerous, existing methods for generating *m*-complete test suites [13] [3] [5] [14] [4] [8] [12] do exactly this and differ only in a type of state identification sequences they add to traversal sequences.

Driven by this observation and the obvious absence of significant progress in solving the longstanding problem of generating *m*-complete test suite, we revisit it in this paper and aim at answering the question whether *m*-complete test suite is irreducible due to the inevitability of the traversal set.

We observe that a considerable part of an *m*-complete test suite is not related to the traversal set itself, but to the test branching when a test has to be repeatedly executed to apply all the sequences of the traversal set. Apparently, the test length reduction can only be achieved by reducing the test branching, which in turn can be obtained by distributing the traversal set over several tests. The caveat is that an arbitrary distribution of the traversal set may break the *m*-completeness of a resulting test suite. Thus, we need first to establish conditions for a distribution of the traversal set such that the *m*-completeness of a test suite is preserved. The main idea developed in this paper is to distribute it among those tests in a test suite which are convergent, i.e., transfer to the same state, in all FSMs of the fault domain which pass the test suite. The approach we elaborate is based on properties of FSM tests, namely their convergence and divergence. We investigate when the convergence and divergence of tests in the specification (which can be easily checked) can be safely assumed to also hold in the implementation under test. The divergence of two tests can be witnessed by different outputs produced by the tests. On the other hand, although convergence of two tests cannot be directly ascertained by considering only the two tests, we show that the knowledge of the maximum number of states of FSMs in the fault domain can be used to formulate conditions for the convergence of tests. We then use the notion of convergence and divergence to state necessary and sufficient conditions for a test suite to be *m*-complete.

Based on these conditions, we elaborate a method, called SPY-method, for *m*-complete test suite generation. The method distributes the sequences of the traversal set over several tests in order to reduce test branching and generate shorter test suites. To assess the potential saving which can be obtained with the approach proposed in this paper, we experimentally compare it with the HSI method [14]. The results suggest that SPY-method can generate test suites up to 40% shorter, on average.

The rest of the paper is organized as follows. In Section 2, we provide the necessary basic definitions. In Section 3, we formally state the problem of generating *m*-complete test suites and discuss existing methods. In Section 4, we investigate test properties and formulate conditions for guaranteeing the *m*-completeness of test suites. In Section 5, we develop a generation method based on the proposed conditions. In Section 6, the method is illustrated on an example. Experimental results are reported in Section 7 and Section 8 concludes the paper.

2. Definitions

A Finite State Machine is a (complete) deterministic Mealy machine, which can be defined as follows.

Definition 1. *A* Finite State Machine (FSM) $\mathcal{S}$ *is a 6-tuple* $(S, s_0, X, Y, \delta_S, \lambda_S)$, *where*

- *S is a finite set of states with the initial state s_0,*
- *X is a finite set of inputs,*
- *Y is a finite set of outputs,*
- *$\delta_S : S \times X \to S$ is a transition function, and*
- *$\lambda_S : S \times X \to Y$ is an output function.*

A tuple $(s, x) \in S \times X$ is a *transition* of $\mathcal{S}$. We extend the transition and output functions from input symbols to input sequences, including the empty sequence ε, as usual: for $s \in S$, $\delta_S(s, \varepsilon) = s$ and $\lambda_S(s, \varepsilon) = \varepsilon$; and for input sequence α and input x, $\delta_S(s, \alpha x) = \delta_S(\delta_S(s, \alpha), x)$ and $\lambda_S(s, \alpha x) = \lambda_S(s, \alpha)\lambda_S(\delta_S(s, \alpha), x)$. An FSM $\mathcal{S}$ is said to be *initially connected*, if for each state $s \in S$, there exists an input sequence $\alpha \in X^*$, called a *transfer* sequence for state s, such that $\delta_S(s_0, \alpha) = s$. In this paper, only initially connected machines are considered. Input sequences *converge* if they are transfer sequences for the same state. Similarly, input sequences *diverge* if they are transfer sequences for different states of the same FSM. A set $K \subseteq X^*$ is a *state cover* for $\mathcal{S}$ if it contains at least one transfer sequence for each state of $\mathcal{S}$. A state cover is *minimal* if it contains exactly one transfer sequence for each state. A set $A \subseteq X^*$ *covers* a transition (s, x) if there exist α, $\alpha x \in A$, where α is a transfer sequence for s. The set A is a *transition cover* for $\mathcal{S}$ if it covers every transition of $\mathcal{S}$. A set of sequences is *initialized*, if it contains the empty sequence.

Given sequences $\alpha, \beta, \gamma \in X^*$, if $\beta = \alpha\gamma$, then α is a *prefix* of β, and γ is a *suffix* of β; if γ is not the empty sequence, then α is a *proper* prefix of β. We also say that a prefix of γ *extends* α (*in* β) and that β is an *extension* of α. We denote by $pref(\beta)$ the set of all prefixes of β. For a set of sequences A, $pref(A)$ is the union of $pref(\beta)$, for all $\beta \in A$. If $A = pref(A)$, then we say that A is *prefix closed*. Given two sets of sequences A and B, we denote by $A.B$ the set of sequences $A.B = \{\alpha\beta \mid \alpha \in A \text{ and } \beta \in B\}$. We will slightly abuse the notation by writing $\alpha.B$ instead of $\{\alpha\}.B$ and $A.\beta$ instead of $A.\{\beta\}$. For a natural number k, we denote by $X^{\leq k}$ the set of all input sequences of length at most k.

Given a set $A \subseteq X^*$, states s and s' are A-*equivalent*, if $\lambda_S(s, \gamma) = \lambda_S(s', \gamma)$ for all $\gamma \in A$. Otherwise, s and s' are A-*distinguishable*. We say that γ distinguishes s and s', if s and s' are $\{\gamma\}$-distinguishable. States s, s' are *equivalent*, if they are X^*-equivalent. Similarly, they are *distinguishable* if they are X^*-distinguishable. We define distinguishability and equivalence of machines as a corresponding relation between their initial states. An FSM is *minimal*, if all states are pairwise distinguishable. In this paper, all the FSMs are assumed to be minimal. A *characterization set* is a set of sequences W such that every two states are W-distinguishable. The set $W_s \subseteq W$ is a *state identifier* for state s if any other state is W_s-distinguishable from s. A *family of harmonized state identifiers* is a collection of sets $\{H_s \mid s \in S\}$, such that states s and s' are $(pref(H_s) \cap pref(H_{s'}))$-distinguishable.

3. Problem Statement and Existing Methods

In this section, we discuss the problem of generating test suites with full fault coverage along with the existing methods and present the main idea of the approach elaborated in this paper.

Henceforth, we assume that $\mathcal{S} = (S, s_0, X, Y, \delta_S, \lambda_S)$ and $\mathcal{Q} = (Q, q_0, X, Y', \delta_Q, \lambda_Q)$ are a specification FSM and an implementation FSM, respectively. Moreover, n is the number of states of $\mathcal{S}$. We denote by $\Im$ the set of all minimal implementation FSMs with the same input alphabet as $\mathcal{S}$. The set $\Im$ is called a *fault domain* for $\mathcal{S}$. For $m \geq n$, let $\Im_m$ be the FSMs of $\Im$ with at most m states, i.e., the set $\Im_m$ represents all faults that can occur in an implementation of $\mathcal{S}$ with at most m states. We denote the maximum number of extra states that an implementation may have by $\Delta = m - n$. Faults can be detected by tests, which are input sequences of the specification FSM $\mathcal{S}$.

Definition 2. *An input sequence of FSM $\mathcal{S}$ is called a* test case *(or simply a* test*) of $\mathcal{S}$. A* test suite *of $\mathcal{S}$ is a finite prefix closed set of tests of $\mathcal{S}$. A test suite T of FSM $\mathcal{S}$ is m-*complete, *if for each FSM $\mathcal{Q} \in \Im_m$, distinguishable from $\mathcal{S}$, there exists a test in T that distinguishes them.*

An FSM *passes* a test suite T if it is T-equivalent to the specification. Thus, a test suite is m-complete if the FSMs in $\Im_m$ which pass it are equivalent to the specification. Two tests α and β in a given test suite T are *T-separable*, if there exist $\alpha\gamma, \beta\gamma \in T$, such that states $\delta_S(s_0, \alpha)$ and $\delta_S(s_0, \beta)$ are $\{\gamma\}$-distinguishable. Clearly, if T-separable tests α and β are convergent in some implementation FSM, it can be distinguished from $\mathcal{S}$ by either $\alpha\gamma$ or $\beta\gamma$.

Since the distinguishability of FSMs is defined as the corresponding relation of their initial states, tests are assumed to be applied in the initial state. For accounting to the reset operation required to bring the FSMs to the initial state, we define the length of a test α as $|\alpha| + 1$, where $|\alpha|$ is the number of input symbols in α. As the application of a test results in the application of all its prefixes, the length of a test suite T, denoted by $len(T)$, is the sum of the lengths of all tests in T which are not proper prefixes of other tests in T.

In this paper, we address the problem of generating an m-complete test suite, when implementation FSMs can have more that n states, i.e., $m \geq n$. This problem has received much less attention compared to the (classical) problem of constructing n-complete test suites, often called checking experiments. One of the main reasons might be the fact that test generation becomes more challenging. To illustrate this, let us consider the FSMs in Figures 1 and 2, where $\mathcal{S}_0$ is the specification machine and $\mathcal{S}_1$ is an implementation machine, which has two extra states. Notice that states 1 and 2 in S_1 are similar to states 1 and 2 in S_0, except that S_1 has two extra states 1' and 2', and the transition (2, *b*) leads to an "erroneous" state 2'.

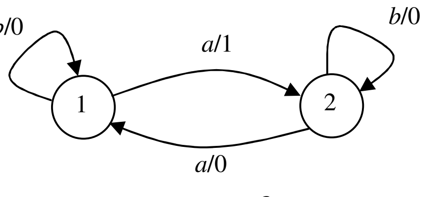


Fig. 1. FSM S_0.

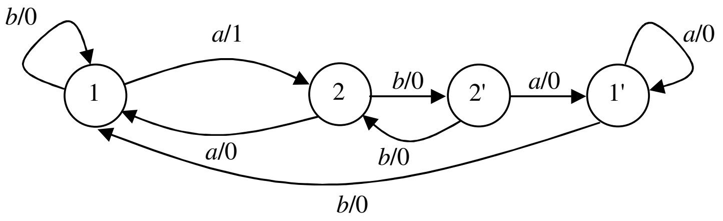


Fig. 2. FSM S_1.

The shortest test able to distinguish S_0 and S_1 should be formed by the input sequence *a*, which is a transfer sequence for state 2, and the input sequence *baa*. Indeed, for any other input sequence of length three, it is possible to construct a distinguishable FSM with two extra states for which only that particular sequence applied to a proper state distinguishes it from S_0. As those FSMs are in the fault domain $\Im_4$, any 4-complete test suite for S_0 should include all input sequences of length three, applied to all states of S_0. In the general case, an *m*-complete test suite for an FSM with *n* states should include all input sequences of length $\Delta + 1$, applied to each state. An early work of Moore [11] uses such sequences to establish a lower bound for sequences identifying ''combination lock'' machines. In fact, the lower bound for the length of an *m*-complete test suite for an FSM with *n* states and *p* inputs is $O(n^3p^{\Delta+1})$, i.e., it is exponential on the number of extra states [13].

Existing methods, such as W [13] [3], Wp [5], HSI [14] and H [4], which generate an *m*-complete test suite *T* for a given minimal deterministic FSM *S* can be summarized as follows.

Step 1: Determine a minimal initialized state cover *K* for *S*.

Step 2: Extend the sequences in *K* by the (traversal) set $X^{\leq\Delta+1}$.

Step 3: Extend the sequences in $K.X^{\leq\Delta+1}$ in such a way that any two divergent sequences, i.e., reaching distinct states in *S*, are *T*-separable.

Existing methods differ mainly in the sequences they use to ensure *T*-separability in Step 3. In the W method, all sequences in $K.X^{\leq\Delta+1}$ are extended by a characterization set. The Wp method uses a characterization set for sequences in $K.X^{\leq\Delta}$ and state identifiers for the other sequences. The HSI method uses the harmonized state identifiers for all sequences in $K.X^{\leq\Delta+1}$. The H method determines on-the-fly a distinguishing sequence for states reached by each pair of divergent sequences in *K*. $X^{\leq\Delta+1}$.

We illustrate the generation of a 3-complete test suite for the 2-state FSM in Figure 1 following the strategy used by the existing methods. For this machine, the characterization set corresponds to a family of harmonized state identifiers $W = H_1 = H_2 = \{a\}$. A minimal state cover for this FSM is $K = \{\varepsilon, a\}$. Then, in the W, Wp, HSI, H methods the sequences in $K.X^{\leq\Delta+1} = pref(\{aaa, aab, aba, abb, ba, bb\})$ are extended by the sequence *a*. The resulting test suite is $T_1 = pref(\{aaaa, aaba, abaa, abba, baa, bba\})$ of length 28; Figure 3 shows its tree representation, where nodes are labelled with states of the specification FSM and edges are labelled with inputs. Each test corresponds to the sequence of inputs along a path from the root to a node.

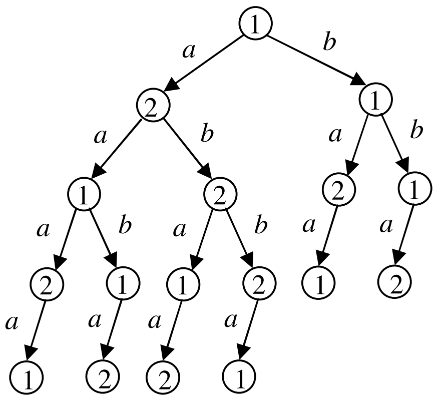
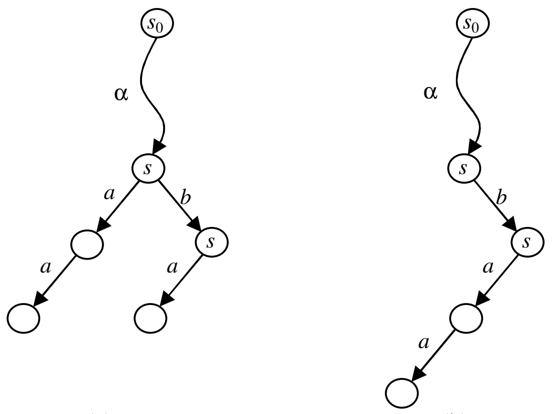


Fig. 3. Tree representation of a 3-complete test suite for S_0.

If, in Step 1, shortest transfer sequences are included into a state cover and, in Step 3, shortest distinguishing sequences are used, tests in a resulting *m*-complete test suite cannot be shortened and if we want to reduce the total length of a test suite we need to find a way of reducing test branching. Indeed, once a test of length *l* branches into *k* tests, the test prefix of *l* inputs contributes *kl* inputs to the total length of a test suite. For instance, each of tests *aa*, *ab* and *b* branches into two tests in T_1, thus contributing twice to its total length. In the existing methods, test branching occurs mainly in Step 2, where each test in a minimal state cover is extended by the sequences in the traversal set $X^{\leq\Delta+1}$. As a result of this, such a test branches into at least $|X|^{\Delta+1}$ tests. Apparently, the test length reduction could be achieved by reducing the test branching, which in turn can be performed by distributing the traversal set $X^{\leq\Delta+1}$ over several tests. As soon as one of these tests is a proper prefix of another the overall test branching and thus the test length are reduced. This key observation is illustrated in Figure 4. Assume that test α should be extended by the sequences *aa* and *ba*. In Figure 4(a) both sequences extend α, branching the test. Consequently, α contributes twice to the length of the test suite. Suppose that tests α and αb are convergent, and, instead of α, the test αb is extended by *aa*, as shown in Figure 4(b). We note that this results in a test suite which is, all things being equal, $|\alpha| - 1$ inputs shorter than before. The problem is that an arbitrary distribution of the traversal set may break the m-completeness of a resulting test suite. Thus, we need first to establish conditions for a distribution of the traversal set $X^{\leq \Delta+1}$ such that the m-completeness of a test suite is preserved. The main idea developed in this paper is to distribute it among those tests in a test suite which are convergent, i.e., transfer to the same state, in all FSMs of the fault domain which pass the test suite, reducing test branching.



(a) (b)

Fig. 4. Test branching (a) *versus* test extension (b).

4. Test Properties

The approach elaborated in this paper is based on properties of FSM tests, namely their convergence and divergence. Recall that two defined input sequences of an FSM converge or diverge if they are transfer sequences for the same state or for different ones, respectively. We generalize these notions to sets of FSMs. Given a non-empty set of FSMs $\Sigma \subseteq \Im$ and two tests $\alpha, \beta \in X^*$, we say that α and β are Σ-*convergent*, if they converge in each FSM of the set Σ. Similarly, we say that α and β are Σ-*divergent*, if they diverge in each FSM of Σ. Two tests are $\mathcal{S}$-convergent ($\mathcal{S}$-divergent) if they are $\{\mathcal{S}\}$-convergent ($\{\mathcal{S}\}$-divergent). Moreover, when it is clear from the context, we will drop the set in which tests are convergent or divergent.

Test convergence and divergence with respect to a single FSM are complementary, i.e., any two tests are either convergent or divergent. However, when a set of FSMs Σ is considered, some tests are neither Σ-convergent nor Σ-divergent. Notice that the Σ-convergence relation is reflexive, symmetric, and transitive, i.e., it is an equivalence relation over the set of tests. Given a test α, let $[\alpha]$ be the corresponding equivalence class in a non-empty set Σ of FSMs with the same input alphabet. The test convergence and divergence possess the following properties.

Lemma 1. *Given tests α, β, such that $[\alpha] = [\beta]$, the following properties hold:*

(i) $[\alpha\gamma] = [\beta\gamma]$, *for any input sequence γ.*

(ii) *For any test φ, if $[\alpha] \neq [\varphi]$, then $[\beta] \neq [\varphi]$.*

An important property of T-separable tests is that they are divergent in all FSMs which are T-equivalent to $\mathcal{S}$. Given a test suite T, let $\Im(T)$ be the set of all $\mathcal{Q} \in \Im$, such that $\mathcal{Q}$ and $\mathcal{S}$ are T-equivalent, i.e., $\Im(T)$ is the set of all FSMs in $\Im$ which pass the test suite T.

Lemma 2. *Given a test suite T of an FSM $\mathcal{S}$, T-separable tests are $\Im(T)$-divergent.*
Proof. Let tests α and β be T-separable. Thus, there exist a sequence γ such that $\alpha\gamma$, $\beta\gamma \in T$ and $\lambda_{\mathcal{S}}(\delta_{\mathcal{S}}(s_0, \alpha), \gamma) \neq \lambda_{\mathcal{S}}(\delta_{\mathcal{S}}(s_0, \beta), \gamma)$. Let $\mathcal{Q}$ be an FSM T-equivalent to $\mathcal{S}$; thus, we have that $\lambda_{\mathcal{S}}(\delta_{\mathcal{S}}(s_0, \alpha), \gamma) = \lambda_{\mathcal{Q}}(\delta_{\mathcal{Q}}(q_0, \alpha), \gamma)$ and $\lambda_{\mathcal{S}}(\delta_{\mathcal{S}}(s_0, \beta), \gamma) = \lambda_{\mathcal{Q}}(\delta_{\mathcal{Q}}(q_0, \beta), \gamma)$. It follows that $\lambda_{\mathcal{Q}}(\delta_{\mathcal{Q}}(q_0, \alpha), \gamma) \neq \lambda_{\mathcal{Q}}(\delta_{\mathcal{Q}}(q_0, \beta), \gamma)$. Thus, $\delta_{\mathcal{Q}}(q_0, \alpha) \neq \delta_{\mathcal{Q}}(q_0, \beta)$. ♦

Existing methods for test generation ensure that two tests are divergent by extending them with an appropriate distinguishing sequence. However, Lemmas 1 and 2 indicate that the convergence and divergence of tests also applies to their equivalence classes. It is thus important to identify under which conditions tests are guaranteed to be convergent, i.e., belong to the same equivalence class.

Ensuring convergence is more involved than ensuring divergence; divergence of two tests can be witnessed by different outputs produced in response to a common suffix sequence. The two tests are thus divergent in any FSM T-equivalent to $\mathcal{S}$. However, convergence of two tests cannot be directly ascertained by considering only the two tests. It turns out that the knowledge of the maximum number of states of FSMs in the fault domain allows us to formulate conditions for the convergence of tests. Given a test suite T and a natural number $m \geq n$, let $\Im_m(T) = \Im_m \cap \Im(T)$, i.e., the set of FSMs in $\Im$ which are T-equivalent to $\mathcal{S}$ and have at most m states.

As $\mathcal{S}$ is in the fault domain $\Im_m(T)$, tests which are $\Im_m(T)$-convergent are also $\mathcal{S}$-convergent. Thus, two tests can be $\Im_m(T)$-convergent only if they are $\mathcal{S}$-convergent.

Definition 3. *A set of tests is* $\Im_m(T)$-convergence-preserving *if all its $\mathcal{S}$-convergent tests are $\Im_m(T)$-convergent. Similarly, a set of tests is* $\Im_m(T)$-divergence-preserving *if all its $\mathcal{S}$-divergent tests are $\Im_m(T)$-divergent.*

In other words, a set of tests is $\Im_m(T)$-convergence-preserving if the convergence in the specification FSM is "preserved" in each FSM which passes the test suite T. Similarly, a set of tests is $\Im_m(T)$-divergence-preserving if the divergence in the specification FSM is preserved in each FSM which passes the test suite T.

In the following lemma, the $\Im_m(T)$-convergence relation is considered; thus, $[\alpha]$ is the subset of tests of T which are $\Im_m(T)$-convergent with test α.

Lemma 3. *Given a test suite T for an FSM $\mathcal{S}$ and $\Delta = m - n \geq 0$, let π and φ be $\mathcal{S}$-convergent tests in T, such that, for any sequence υ of length Δ, there exist tests $\alpha \in [\pi]$, $\beta \in [\varphi]$, and an $\Im_m(T)$-divergence-preserving state cover for $\mathcal{S}$ in T containing $\{\alpha, \beta\}.pref(\upsilon)$. Then, π and φ are $\Im_m(T)$-convergent.*
Proof. Suppose that π and φ are not $\Im_m(T)$-convergent. Thus, there exists $\mathcal{Q} \in \Im_m(T)$, such that π and φ are $\mathcal{Q}$-divergent. As π and φ are $\mathcal{S}$-convergent, the FSM $\mathcal{Q}$ is not

equivalent to $\mathcal{S}$ and there must exist an input sequence γ such that $\mathcal{S}$ and $\mathcal{Q}$ are $\{\pi\gamma, \varphi\gamma\}$-distinguishable. Assume that γ is a shortest input sequence with this property. Thus,

$$\mathcal{S} \text{ and } \mathcal{Q} \text{ are } (([\pi] \cup [\varphi]).\gamma')\text{-equivalent, for all } \gamma', \text{ such that } |\gamma'| < |\gamma|. \qquad (1)$$

We have that $|\gamma| > \Delta$, since otherwise there would exist $\alpha' \in [\pi]$ and $\beta' \in [\varphi]$ such that $\{\alpha'\gamma, \beta'\gamma\} \subseteq T$, implying that $\mathcal{S}$ and $\mathcal{Q}$ are T-distinguishable.

Let $\alpha \in [\pi]$ and $\beta \in [\varphi]$ be such that there exists an $\Im_m(T)$-divergence-preserving state cover for $\mathcal{S}$ in T containing the set $\{\alpha, \beta\}.pref(\gamma_\Delta)$, where γ_i is the prefix of γ of length i. Without loss of generality, we assume that $\mathcal{S}$ and $\mathcal{Q}$ are $\{\alpha\gamma\}$-distinguishable, i.e., $\lambda_\mathcal{Q}(q_0, \alpha\gamma) \neq \lambda_\mathcal{S}(s_0, \alpha\gamma)$. Let $A_i = \{\alpha, \beta\}.pref(\gamma_i)$, $0 \leq i \leq \Delta$. The tests $\alpha\gamma_i$ and $\beta\gamma_i$ are $\mathcal{Q}$-divergent and, moreover, A_i is $\Im_m(T)$-divergence-preserving. We show by induction that, for all $0 \leq i \leq \Delta$, $|\delta_\mathcal{Q}(q_0, A_i)| \geq i + |\delta_\mathcal{S}(s_0, A_i)| + 1$.

Base case: For $i = 0$, we have that $A_0 = \{\alpha, \beta\}$. As α and β are $\mathcal{S}$-convergent and $\mathcal{Q}$-divergent, the result follows, since $|\delta_\mathcal{Q}(q_0, A_0)| = 2$ and $|\delta_\mathcal{S}(s_0, A_0)| = 1$.

Inductive Step: Suppose that the result holds i, $0 \leq i < \Delta$, i.e.,

$$|\delta_\mathcal{Q}(q_0, A_i)| \geq i + |\delta_\mathcal{S}(s_0, A_i)| + 1. \qquad (2)$$

We show that the result holds for $i + 1$. Let $j \leq i$. Suppose that $\alpha\gamma_{i+1}$ and $\alpha\gamma_j$ are $\mathcal{S}$-divergent; then $\alpha\gamma_{i+1}$ is $\mathcal{Q}$-divergent with $\alpha\gamma_j$ and $\beta\gamma_j$, since A_{i+1} is $\Im_m(T)$-divergence-preserving. Suppose now that $\alpha\gamma_{i+1}$ and $\alpha\gamma_j$ are $\mathcal{S}$-convergent. Let χ be the suffix which extends γ_{i+1} in γ, i.e., $\gamma = \gamma_{i+1}\chi$. If $\alpha\gamma_{i+1}$ is $\mathcal{Q}$-convergent with $\alpha\gamma_j$, then $\alpha\gamma_j\chi$ distinguishes $\mathcal{S}$ and $\mathcal{Q}$, since $\lambda_\mathcal{Q}(q_0, \alpha\gamma_j\chi) = \lambda_\mathcal{Q}(q_0, \alpha\gamma_{i+1}\chi) = \lambda_\mathcal{Q}(q_0, \alpha\gamma) \neq \lambda_\mathcal{S}(s_0, \alpha\gamma) = \lambda_\mathcal{S}(s_0, \alpha\gamma_{i+1}\chi) = \lambda_\mathcal{S}(s_0, \alpha\gamma_j\chi)$. As $|\gamma_j\chi| < |\gamma_{i+1}\chi| = |\gamma|$, it follows that $\alpha\gamma_{i+1}$ should be $\mathcal{Q}$-divergent with $\alpha\gamma_j$ and $\beta\gamma_j$, since otherwise we have a contradiction to (1). By the same token, the test $\alpha\gamma_{i+1}$ is $\mathcal{Q}$-divergent with $\beta\gamma_j$. Thus, $\alpha\gamma_{i+1}$ is $\mathcal{Q}$-divergent with $\alpha\gamma_j$, $j \leq i$, i.e., with all tests in A_i and reaches a state in $\mathcal{Q}$ which is not reached by the tests in A_i. Hence,

$$|\delta_\mathcal{Q}(q_0, A_{i+1})| \geq |\delta_\mathcal{Q}(q_0, A_i)| + |\delta_\mathcal{Q}(q_0, \alpha\gamma_{i+1})| \geq |\delta_\mathcal{Q}(q_0, A_i)| + 1. \qquad (3)$$

If $\alpha\gamma_{i+1}$ is $\mathcal{S}$-convergent with some test in A_i, then

$$|\delta_\mathcal{S}(s_0, A_{i+1})| = |\delta_\mathcal{S}(s_0, A_i)|. \qquad (4)$$

The induction thus applies, since

$$|\delta_\mathcal{Q}(q_0, A_i)| \geq i + |\delta_\mathcal{S}(s_0, A_i)| + 1 \qquad \text{(inductive hypothesis (2))}$$
$$|\delta_\mathcal{Q}(q_0, A_i)| + 1 \geq (i + 1) + |\delta_\mathcal{S}(s_0, A_i)| + 1$$
$$|\delta_\mathcal{Q}(q_0, A_{i+1})| \geq (i + 1) + |\delta_\mathcal{S}(s_0, A_{i+1})| + 1 \qquad \text{(due to (3) and (4))}$$

On the other hand, if $\alpha\gamma_{i+1}$ is $\mathcal{S}$-divergent with all tests in A_i, then

$$|\delta_\mathcal{S}(s_0, A_{i+1})| = |\delta_\mathcal{S}(s_0, A_i)| + 1 \qquad (5)$$

In this case, $\beta\gamma_{i+1}$ is also $\mathcal{Q}$-divergent with all tests in A_i, since A_{i+1} is $\Im_m(T)$-divergence-preserving. Moreover, $\beta\gamma_{i+1}$ is $\mathcal{Q}$-divergent with $\alpha\gamma_{i+1}$. Thus, we have that

$$|\delta_\mathcal{Q}(q_0, A_{i+1})| = |\delta_\mathcal{Q}(q_0, A_i)| + |\delta_\mathcal{Q}(q_0, \{\alpha\gamma_j, \beta\gamma_j\})| \geq |\delta_\mathcal{Q}(q_0, A_i)| + 2 \qquad (6)$$

The induction also applies, since

$$|\delta_\mathcal{Q}(q_0, A_i)| \geq i + |\delta_\mathcal{S}(s_0, A_i)| + 1 \qquad \text{(inductive hypothesis (2))}$$
$$|\delta_\mathcal{Q}(q_0, A_i)| + 2 \geq (i + 1) + (|\delta_\mathcal{S}(s_0, A_i)| + 1) + 1$$
$$|\delta_\mathcal{Q}(q_0, A_{i+1})| \geq (i + 1) + |\delta_\mathcal{S}(s_0, A_{i+1})| + 1 \qquad \text{(due to (5) and (6))}$$

This concludes the induction proof. Then, for all $0 \leq i \leq \Delta$, it holds that $|\delta_\mathcal{Q}(q_0, A_i)| \geq i + |\delta_\mathcal{S}(s_0, A_i)| + 1$. In particular, the set of tests A_Δ reaches at least $\Delta + |\delta_\mathcal{S}(s_0, A_\Delta)| + 1$ states in $\mathcal{Q}$.

Consider now a smallest set K, such that $K \cup A_\Delta$ is an $\Im_m(T)$-divergence-preserving state cover for $\mathcal{S}$ in T; thus, $|K| = n - |\delta_S(s_0, A_\Delta)|$, since α and β are $\mathcal{S}$-convergent. As $K \cup A_\Delta$ is $\Im_m(T)$-divergence-preserving, the tests of the set K reach exactly $n - |\delta_S(s_0, A_\Delta)|$ states in $\mathcal{Q}$, and each of them is distinct from all states reached by A_Δ. Thus, the tests in $K \cup A_\Delta$ reach at least $n - |\delta_S(s_0, A_\Delta)| + \Delta + |\delta_S(s_0, A_\Delta)| + 1 = n + m - n + 1 = m + 1$ states in $\mathcal{Q}$, contradicting the fact that $\mathcal{Q}$ has at most m states. ♦

The importance of Lemma 3 for test generation is that it shows how to ensure the $\Im_m(T)$-convergence of two $\mathcal{S}$-convergent tests. This in turn, allows including these tests into the same equivalence class. Then, Lemma 1 can be applied, which indicates that if a test should be extended by given sequences, e.g., from the traversal set, any tests of its equivalence class can be chosen, distributing these sequences over several tests. Lemma 3 also leads to the necessary and sufficient conditions for test completeness with respect to the fault domain $\Im_m$, where each FSM has at most m states, $m \geq n$.

Theorem 1. *Let T be a test suite for an FSM $\mathcal{S}$ with n states and $m \geq n$. Then, the following statements are equivalent:*
(i) T is an m-complete test suite for $\mathcal{S}$
(ii) T contains an $\Im_m(T)$-convergence-preserving initialized transition cover for $\mathcal{S}$.
Proof.

(ii) $\Rightarrow$ (i) Let T contain an $\Im_m(T)$-convergence-preserving initialized transition cover A for $\mathcal{S}$, and $\mathcal{Q} \in \Im_m(T)$. Define the relation $h \subseteq S \times Q$ as follows:

$$(s, q) \in h \Leftrightarrow \text{there exists } \alpha \in A, \text{ such that } \delta_S(s_0, \alpha) = s \text{ and } \delta_Q(q_0, \alpha) = q.$$

As A is a transition cover for $\mathcal{S}$, for each $s \in S$ there exists $q \in Q$ such that $(s, q) \in h$. Moreover, as A is $\Im_m(T)$-convergence-preserving, for each $s \in S$, there exists only one $q \in Q$ such that $(s, q) \in h$; thus, h is a mapping. As $\varepsilon \in A$,

$$h(s_0) = q_0.$$

Let $s \in S$ and $x \in X$. As A is a transition cover for $\mathcal{S}$,

$$\text{there exists } \alpha x \in A \text{ such that } \delta_S(s_0, \alpha) = s.$$

Correspondingly,

$$h(\delta_S(\delta_S(s_0, \alpha), x) = h(\delta_S(s_0, \alpha x)) = \delta_Q(q_0, \alpha x) = \delta_Q(\delta_Q(q_0, \alpha), x) = \delta_Q(h(\delta_S(s_0, \alpha)), x)$$

and

$$\lambda_S(\delta_S(s_0, \alpha), x) = \lambda_Q(\delta_Q(q_0, \alpha), x) = \lambda_Q(h(\delta_S(s_0, \alpha)), x),$$

as $\mathcal{Q} \in \Im_m(T)$.

Thus, h is an isomorphism and, as $h(s_0) = q_0$, it follows that $\mathcal{Q}$ is equivalent to $\mathcal{S}$.

(i) $\Rightarrow$ (ii) Let T be an m-complete test suite. First, notice that any m-complete test suite is a transition cover for the FSM $\mathcal{S}$. Otherwise, there exists a transition of $\mathcal{S}$ which is not traversed by the test suite; an FSM that is T-equivalent to, but distinguishable from, $\mathcal{S}$ can be obtained from $\mathcal{S}$ by mutating the output in this transition. By definition, T is prefix closed, thus, it is an initialized transition cover.

As T is an m-complete test suite, each FSM $\mathcal{Q} \in \Im_m(T)$ is equivalent to $\mathcal{S}$, i.e., there exists a mapping h: $S \to Q$ such that $h(s_0) = q_0$ and for each transition (s, x) it holds that

$$h(\delta_S(s, x)) = \delta_Q(h(q), x)$$

and thus, since $h(s_0) = q_0$, for each input sequence α it holds that

$$h(\delta_S(s_0, \alpha)) = \delta_Q(h(s_0), \alpha) = \delta_Q(q_0, \alpha).$$

Let α and β be $\mathcal{S}$-convergent, i.e., $\delta_S(s_0, \alpha) = \delta_S(s_0, \beta)$. It follows that

$$\delta_Q(q_0, \alpha) = h(\delta_S(s_0, \alpha)) = h(\delta_S(s_0, \beta)) = \delta_Q(q_0, \beta).$$

Thus, α and β are also $\mathcal{Q}$-convergent and, consequently, the set is $\Im_m(T)$-convergence-preserving.♦

Considering the generation methods discussed in Section 3, we note that the conditions of Lemma 3 are satisfied for all pairs of $\mathcal{S}$-convergent tests in $K \cup K.X$, which turns out to be a transition cover for $\mathcal{S}$. Thus, the test suites generated by these methods satisfy the conditions of Theorem 1, since $K \cup K.X$ is an initialized transition cover. At the same time, Theorem 1 suggests that rather than considering the whole set of tests in $K.X^{\leq\Delta+1}$ at once, as the existing methods do, it is sufficient to ensure convergence of tests covering all transitions, using Lemma 3. Moreover, Lemmas 1, 2, and 3 indicate that this can be achieved in an iterative way, namely, the convergence for tests covering a current transition can be ensured based on the convergence established for other transitions. In the next section we elaborate this idea in a method for complete test suite generation.

5. Test Generation Method

In this section, we present a method, called SPY-method, which generates an m-complete test suite by building an $\Im_m(T)$-convergence-preserving transition cover. In the method, the knowledge about test convergence and divergence obtained during the execution helps identify the possibility of extending tests already in the test suite. Such an extension avoids branching of tests and thus contributes to test suite shortening. During the execution of the method, the $\Im_m(T)$-convergence of tests is determined. Notice that any two $\Im_m(T)$-convergent tests are also $\Im_m(T')$-convergent, for each $T' \supseteq T$. Thus, the inclusion of new tests in T does not invalidate this property.

As the $\Im_m(T)$-convergence relation is an equivalence relation, it can be represented by the partition it induces. In a given stage of the method execution only a subset of the $\Im_m(T)$-convergence relation might be known. We denote by Π the partition induced by the pairs of tests which are known to be $\Im_m(T)$-convergent. Given a test $\alpha \in T$, we denote by $[\alpha]_\Pi$ the block of the partition Π which contains α.

We assume that a family H of harmonized state identifiers is provided. Given a test α, let $H(\alpha) \in H$ be the state identifier for state $\delta_S(s_0, \alpha)$. The method starts by determining a minimal initialized state cover K, as in Step 1 of existing methods. Then, the tests in K are extended by the appropriate state identifiers. Each block in the initial partition Π is a singleton, since no convergence is initially known. The method iterates until the set of tests which are $\Im_m(T)$-convergent with the tests in K becomes a transition cover for $\mathcal{S}$.

During the execution of the method, it is necessary to extend two tests in T to ensure their divergence. As the divergence of tests also applies to other tests in their blocks, when more than one test is available in a given block, the one which will result in a shorter test suite is selected. This is achieved as follows. Suppose that test $\alpha \notin T$ should be added to T. Let β be the longest prefix of α which is in T. If β is not a proper prefix of another test in T, we have that $len(T \cup \{\alpha\}) = len(T) + |\alpha| - |\beta|$, i.e., adding α to T results only in extending the test β by $|\alpha| - |\beta|$ input symbols. On the other hand, if β is a proper prefix of some other test in T, it holds that $len(T \cup \{\alpha\}) = len(T) + |\alpha| + 1$, as it results in an additional testing branching. Thus, selection of a test which has to be extended by some input sequence, e.g., a state distinguishing sequence, should result, whenever possible, in extending some test in T that is not a proper prefix of another test.

After adding new tests two blocks containing tests that are $\Im_m(T)$-convergent, are merged, i.e., replaced by their union, iteratively. The merge of blocks can result in a new partition for which the $\Im_m(T)$-convergence of other tests can be concluded, due to the application of Lemma 1(i) and thus, the procedure of merging should be repeated. We denote by $closure(\Pi)$ the partition obtained after merging the blocks of Π as much as possible, by applying subset merging and Lemma 1(i).

We now present SPY-method.

```
Input: An FSM $\mathcal{S}$ with $n$ states, a family of harmonized state identifiers $H$ and a
natural number $m \geq n$.
Output: An $m$-complete test suite.
Determine a minimal initialized state cover $K$.
$T := pref(\{\alpha.H(\alpha) \mid \alpha \in K\})$
$\Pi := \{\{\alpha\} \mid \alpha \in T\}$
While there exists a transition $(s, x)$ not covered by the set of tests in $T$ which are
$\Im_m(T)$-convergent with some test in $K$
        Let $\alpha, \beta \in K$ be such that $\delta_S(s_0, \alpha) = s$ and $\delta_S(s_0, \beta) = \delta_S(s, x)$
        For each $\gamma \in X^{\leq\Delta}$, each $\sigma \in H(\beta\gamma)$
                Select $\alpha' \in [\alpha]_\Pi$, such that $len(T \cup \{\alpha' x\gamma\sigma\})$ is minimal
                $T := T \cup pref(\alpha' x\gamma\sigma)$
                Select $\beta' \in [\beta]_\Pi$, such that $len(T \cup \{\beta'\gamma\sigma\})$ is minimal
                $T := T \cup pref(\beta'\gamma\sigma)$
                $\Pi := closure(\Pi \cup \{\{\chi\} \mid \chi \in \{\alpha' x, \beta'\}.pref(\gamma\sigma)\})$
        End for
        $\Pi := closure(\Pi \cup \{[\alpha x]_\Pi \cup [\beta]_\Pi\})$
End while
Return $T$
```

Theorem 2. *SPY-method generates an m-complete test suite T for $\mathcal{S}$.*

Proof. Let $C = \{\beta \in [\alpha]_\Pi \mid \alpha \in K\}$, i.e., C is the set of tests which are $\Im_m(T)$-convergent with some test in K. Notice that C is $\Im_m(T)$-convergence-preserving, since by its definition, any two tests in C which are $\mathcal{S}$-convergent are also $\Im_m(T)$-convergent. We first show that in each iteration of the method, C is extended to cover a transition (s, x) which was not yet covered. Let $\alpha, \beta \in K$ be such that $\delta_S(s_0, \alpha) = s$ and $\delta_S(s_0, \beta) = \delta_S(s, x)$. The method then uses the state identifiers required to ensure that tests αx and β are $\Im_m(T)$-convergent. Indeed, for all $\gamma \in X^{\leq\Delta}$, tests α' and β', which are $\Im_m(T)$-convergent with α and β, respectively, are selected and the tests $\alpha' x\gamma$ and $\beta'\gamma$ are extended with the corresponding state identifiers. As the state cover K is also extended by the state identifiers, we have that for each sequence γ of length Δ, the set $K \cup \{\alpha x, \beta\}.pref(\gamma)$ is $\Im_m(T)$-divergence-preserving; thus, the conditions of Lemma 3 are satisfied and tests αx and β are $\Im_m(T)$-convergent. The the blocks containing αx and β are merged. As a result, the transition (s, x) is covered by C. When the method terminates, C is a transition cover for $\mathcal{S}$. As K is initialized and $K \subseteq C$, C is also initialized. Hence, by Theorem 1, T is m-complete, since $C \subseteq T$ is an $\Im_m(T)$-convergence-preserving initialized transition cover. ♦

In each iteration the proposed method deals with the set $X^{\leq\Delta}$, while the theoretical results indicate that an m-complete test suite should include all sequences in the traversal set $X^{\leq\Delta+1}$ [11] [8]. Notice, however, that to obtain a transition cover as required by Theorem 1, the tests of a state cover has to be extended by X, which is in its turn extended by $X^{\leq\Delta}$. Therefore, all sequences in the traversal set $X^{\leq\Delta+1}$ are indeed present in the resulting test suite. Nevertheless, the distribution of the traversal set over several tests usually results in shorter test suites, as demonstrated by the example and the experimental results on the next sections.

Compared with the existing methods for m-complete test suite generation, SPY-method requires the additional operations of handling the partitions of tests and selecting the tests in a partition which lead to a minimal length increase. We discuss the overhead imposed by these operations. The partitions used in the method can be efficiently handled using a *union-find* structure [6]. The operation of merging blocks and determining to which block a test belongs can be performed in $O(Ack^{-1}(l, l))$, where $Ack^{-1}(l, l)$ is the inverse of the extremely quickly-growing Ackermann function [6]. For any reasonable value of l, $Ack^{-1}(l, l)$ is less than five, i.e., the running time of these operations is effectively a small constant. In order to efficiently calculate a length increase caused by new tests, test suites can be represented by trees. Then it is possible to identify when a test will create a new test (branching at a non-leaf node) or extend an existing one (extending a leaf node) by retrieving the information about nodes in the tree. As the size of the tree is proportional to the length of the test suite, the overhead imposed by the additional operations required by the method, i.e., maintaining the partitions and determining the length increase, is polynomial in the length of the test suite.

6. Example

In this section, we illustrate the execution of the method. Consider the FSM in Figure 1. We generate a 3-complete test suite, using the family of harmonized state identifiers as in Section 3, $H_1 = H_2 = \{a\}$. Note that as before, $n = 2$ and $m = 3$.

The method determines a minimal initialized state cover $K = \{\varepsilon, a\}$. The test suite is initialized with $T := \{\alpha.H(\alpha) \mid \alpha \in K\} = pref(aa)$ and the partition $\Pi := \{\{\varepsilon\}, \{a\}, \{aa\}\}$. Notice that the tests in K already cover the transition $(1, a)$. Then, the method iterates until all the other transitions are also covered by the tests which are $\Im_m(T)$-convergent with either ε or a. Notice that in this example, both H_1 and H_2 contain only the sequence a. Therefore, each state identifier used in the method is always equal to $\{a\}$, i.e., $\sigma = a$ throughout this example.

The method selects the transition $(s, x) = (1, b)$; thus $\alpha = \beta = \varepsilon$. At this stage each block is a singleton; thus, selecting the empty sequence ε is the only option in the first iteration. For each $\gamma \in X^{\leq\Delta} = \{\varepsilon, a, b\}$, the test ε is extended by $x\gamma\sigma$ and $\gamma\sigma$; namely, the empty sequence is extended by the sequences *ba*, *a*, *baa*, *aa*, *bba*, and *ba*. The test suite becomes $T = pref(\{aa, baa, bba\})$ and the partition Π is updated to include the new tests (each of them also becomes a singleton block in the partition). According to Lemma 3, ε and b are now $\Im_m(T)$-convergent, thus, blocks $\{\varepsilon\}$ and $\{b\}$ should be merged. After updating the partition and determining its closure, the partition $\Pi = \{\{\varepsilon, b, bb\}, \{a, ba, bba\}, \{aa, baa\}\}$ is obtained. The resulting test suite is represented in Figure 4. The nodes with the same color are in the same block of the partition Π.

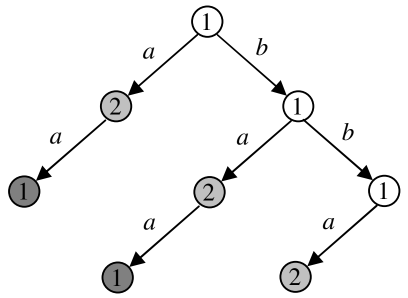


Fig. 4. Tree representation of $pref(\{aa, baa, bba\})$.

The methods selects the transition $(s, x) = (2, a)$. Then $\alpha = a$ and $\beta = \varepsilon$. In this iteration, the blocks of the partition contain several tests; thus, there are choices when selecting the test which is extended by the state identifier. For each $\gamma \in X^{\leq\Delta} = \{\varepsilon, a, b\}$, some test in $[\alpha]_\Pi = [a]_\Pi$ should be extended by $x\gamma\sigma$ and some test in $[\beta]_\Pi = [\varepsilon]_\Pi$ should be extended by $\gamma\sigma$. For $\gamma = \varepsilon$ some test in $[a]_\Pi = \{a, ba, bba\}$ has to be extended by $x\gamma\sigma = aa$; the test suites resulting from extending *a*, *ba* and *bba* by *aa* have lengths 12, 12 and 13, respectively. Thus, the test *a* is selected and *aaa* is added to T. Then, some test in $[\varepsilon]_\Pi = \{\varepsilon, b, bb\}$ should be extended by *a*. As $a \in T$, no additional test is included. For $\gamma = a$, some test in $[a]_\Pi$ has to be extended by $x\gamma\sigma = aaa$ and some test in $[\varepsilon]_\Pi$ by $\gamma\sigma = aa$. A test suite of shorter length can be obtained by extending either *a* or *ba*. The test *a* is selected and *aaaa* is added to T. There is no need to extend any sequence in $[\varepsilon]_\Pi$ by $\gamma\sigma = aa$, since $aa \in T$. For $\gamma = b$, $\sigma = a$, some test in $[a]_\Pi$ should be extended by $x\gamma\sigma = aba$ and some test in $[\varepsilon]_\Pi$ by $\gamma\sigma = ba$. Extending tests *a*, *ba* and *bba* by *aba* results in test suites of lengths 17, 15 and 16, respectively. The test *ba* is, then, selected and *baaba* is added to T. Again, there is no need to extend any sequence in $[\varepsilon]_\Pi$ by $\gamma\sigma$ = *ba*. The test suite becomes T = *pref*({*aaaa*, *baaba*, *bba*}). The tests ε and *aa* are now $\Im_m(T)$-convergent and thus, blocks {ε, *b*, *bb*} and {*aa*, *baa*} should be merged. After merging these blocks and deriving the closure, the partition Π = {{ε, *aa*, *aaaa*, *b*, *baa*, *baab*, *bb*}, {*a, aaa*, *ba*, *baaba*, *bba*}} is obtained. Figure 5 represents the resulting test suite.

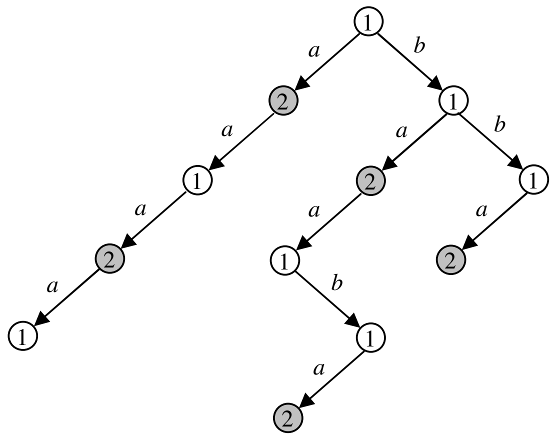


Fig. 5. Tree representation of *pref*({*aaaa*, *baaba*, *bba*}).

It remains to cover the transition $(s, x) = (2, b)$; thus $\alpha = \beta = a$. For $\gamma = \varepsilon$, some test in $[a]_\Pi$ = {*a, aaa*, *ba*, *baaba*, *bba*} should be extended by $x\gamma\sigma$ = *ba* and $\gamma\sigma$ = *a*. The test suites obtained by extending either test *baaba* or *bba* by *ba* have the same length; the test *bba* is then selected and *bbaba* is added to T. Some test in $[a]_\Pi$ has to be extended by $\gamma\sigma$ = *a*, which does not need any additional test, since *aa* ∈ T. For γ = *a*, some test in $[a]_\Pi$ should be extended by $x\gamma\sigma$ = *baa* and $\gamma\sigma$ = *aa*. The test suite of a shorter length is obtained by extending *bba* by *baa* and the test *bbabaa* is added to T. There is no need to extend any test in $[a]_\Pi$ by *aa*, since *aaa* ∈ T. For γ = *b*, some test in $[a]_\Pi$ should be extended by $x\gamma\sigma$ = *bba* and $\gamma\sigma$ = *ba*. The test suite of a shorter length is obtained by extending *baaba* by *bba* and the test *baababba* is added to T. To extend some test in $[a]_\Pi$ by $\gamma\sigma$ = *ba*, no additional test is required, since *baaba* ∈ T and *baa* ∈ $[a]_\Pi$. The resulting test suite is T = *pref*({*aaaa*, *baababba*, *bbabaa*}) of length 21. Recall that the test suite T_1 obtained by the existing methods for the machine in Figure 1 has length 28.

7. Experimental Results

In this section, we present the results of an experiment with the HSI method and the proposed method, comparing the length of the test suites they generate. We randomly generate minimal FSMs with five inputs, five outputs and the number of states n ranging from five to 50. We executed both the HSI method and the proposed method for generating m-complete test suites, for $n \le m \le n + 3$ and calculated the ratio of reduction, i.e., the average ratio of the length of the test suite generated by SPY-method and the length of the test suite generated by the HSI method. For each setting (values of n and m), we generated 30 FSMs and the respective test suites, totalling 5520 FSMs. In Figure 6, we plot the variation of the average ratio with respect to the number of states. We notice that the test suites generated by our method are on average up to 40% shorter than the test suites obtained by the HSI method; moreover, the larger the number of states in the specification FSM and the number of extra states in implementations, the bigger the reduction.

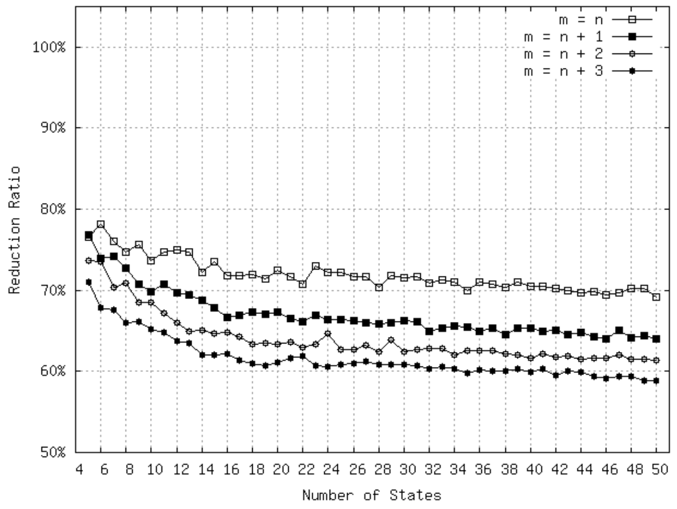


Fig. 6. Average reduction ratio.

8. Conclusions

In this paper we investigated the problem of generating m-complete test suites for an FSM with n states, when implementation FSMs may have extra states.

The main contributions of this paper are as follows. Firstly, although we have not refuted the inevitability of including the sequences of a traversal set in an m-complete test suite, we showed that these sequences can be arranged in such a way that test branching is significantly reduced. Secondly, we stated conditions which guarantee that the resulting test suite is indeed m-complete and elaborated a test generation method based on these conditions. Differently from all existing methods, the proposed method distributes the sequences in the traversal set over several tests avoiding as much as possible test branching and thus leading to shortening of the resulting test suite. Finally, we experimentally compared the proposed method with the HSI-method. The experimental results indicate that obtained tests are on average up to 40% shorter.

As future work, it is possible to combine the on-the-fly determination of distinguishing sequences used in the H method with the possibility of distributing them. Another possible extension is the further investigation of properties of test convergence and divergence.

Acknowledgements

The authors acknowledge financial supports of NSERC (Grant OGP0194381), Brazilian Funding Agency CNPq (Grant 200032/2008-9), and FCP Russian program (contract 02.514.12.4002).

References

1. Bochmann, G. v., Petrenko, A.: Protocol testing: review of methods and relevance for software testing. In: ACM International Symposium on Software Testing and Analysis (ISSTA'94), pp. 109--124. ACM Press, New York (1994)
2. Binder, R.: Testing Object-Oriented Systems. Addison-Wesley, Inc. (2000)
3. Chow, T. S.: Testing software design modeled by finite-state machines. IEEE Trans. Softw. Eng. **4**(3), pp. 178--187 (1978)
4. Dorofeeva, R., El-Fakih, K. and Yevtushenko, N.: An improved conformance testing method. In: Formal Techniques for Networked and Distributed Systems, LNCS, vol. 3731, pp. 204--218. Springer, Heidelberg (2005)
5. Fujiwara, S., von Bochmann, G., Khendek, F., Amalou, M., and Ghedamsi, A.: Test selection based on finite state models. IEEE Trans. Softw. Eng. **17**(6), pp. 591--603 (1991)
6. Galil, Z. and Italiano, G. F.: Data structures and algorithms for disjoint set union problems. ACM Comput. Surv. **23**(3), pp. 319--344 (1991)
7. Gonenc, G.: A method for the design of fault detection experiments. IEEE Trans. on Comput., **19**(6), pp. 551--558 (1970)
8. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines - a survey. In: Proceedings of the IEEE. **84**(8), pp. 1090--1123, IEEE Press, New York (1996)
9. Hennie, F.C.: Fault-detecting experiments for sequential circuits. In: Proceedings of Fifth Annual Symposium on Circuit Theory and Logical Design, pp. 95--110. Princeton, New Jersey (1965)
10. Hierons, R.M., Ural, H.: Optimizing the length of checking sequences. IEEE Trans. on Comput, **55**(5), pp. 618--629 (2006)
11. Moore, E.F.: Gedanken-experiments on sequential machines. Automata Studies, Annals of Mathematical Studies (34), pp. 129--153. (1956)
12. Petrenko, A., Higashino, T., Kaji, T.: Handling redundant and additional states in protocol testing. In: IFIP 8th Inter. Workshop on Protocol Test Systems, pp. 307--322. Chapman & Hall (1995)
13. Vasilevskii, M. P.: Failure diagnosis of automata. Cybernetics, **4**, pp. 653--665 (1973)
14. Yevtushenko, N. and Petrenko, A.: Synthesis of test experiments in some classes of automata. In: Automatic Control and Computer Sciences, **24**(4), pp. 50--55 (1990)