Generating Checking Sequences for Partial Reduced Finite State Machines

Adenilso Simão [1] and Alexandre Petrenko [2]
[1] Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
São Carlos, São Paulo, Brazil
[2] Centre de recherche informatique de Montreal (CRIM)
Montreal, Quebec, Canada
adenilso@icmc.usp.br, petrenko@crim.ca

Abstract. The problem of generating checking sequences for FSMs with distinguishing sequence has been attracting interest of researchers for several decades. In this paper, a solution is proposed for partial reduced FSMs with distinguishing sets, and either with or without reset feature. Sufficient conditions for a sequence to be a checking sequence for such FSMs are formulated. Based on these conditions, a method to generate checking sequence is elaborated. The results of an experimental comparison indicate that the proposed method produces shorter checking sequences than existing methods in most cases. The impact of using the reset feature on the length of checking sequence is also experimentally evaluated.

1 Introduction

Test generation from a Finite State Machine (FSM) is an active research topic with numerous contributions over decades, starting with the seminal work of Moore [9] and Hennie [5]. Given a specification FSM *M* and a black box implementation *N*, the objective is to construct a test suite that can check whether *N* behaves correctly according to *M*. It is usual to assume that *N* behaves like an unknown FSM with the same input alphabet and has at most as many states as *M*.

A checking sequence is an input sequence that can be used to check the correctness of the implementation. Many methods for generating checking sequences have been proposed, e.g., [5], [8], [4], [2], [11], [6], [3], [10], [7], and [12]. The crucial issue for these methods is how to guarantee that a black box implementation is in a known state after the application of some input sequence. This problem is somewhat simplified if the specification FSM *M* has a distinguishing sequence, that is an input sequence for which different states of *M* produce different outputs. However, not every FSM has a distinguishing sequence. A distinguishing set is a set of input sequences, one for each state of *M*, such that for each pair of distinct states, the respective input sequences have a common prefix for which both states produce different outputs. A distinguishing set can be obtained from a distinguishing sequence. Notice, however, that there exist FSMs with a distinguishing set which do not have distinguishing sequence [2].

Several generation methods have been proposed for generating checking sequence when a distinguishing sequence is available, e.g., [5], [4], [11], [6], [3], [10], [7], and

[12]. In [5], Hennie elaborates the basis for the existing generation methods. Hennie divides the checking sequence into two parts: in the first part, the distinguishing sequence and some (possibly empty) transfer sequence are applied to each state, verifying that the distinguishing sequence is also a distinguishing sequence for the implementation under test, while in the second part, each transition is verified certifying that the source and target states are correctly implemented. A graph-theoretical method for generation of checking sequence is presented in [4]. No attempt to optimize the checking sequence is made, though. Recently, interest in improving methods for generating checking sequence with distinguishing sequence has revived again. Ural *et al.* [11] state an important theorem with sufficient conditions for a sequence to be a checking sequence for a complete FSM with a distinguishing sequence. A graph-theoretical method is also suggested, which casts the problem of finding a checking sequence as a Rural Chinese Postman Problem, following Aho *et al.*'s work [1]. This work has later been improved in [6] and [7], which fine-tune the modeling of the problem; Chen *et al.* [3] demonstrate that it is sufficient to consider only a subset of the FSM transitions; and Ural and Zhang [10] use the overlapping of the distinguishing sequence with itself to shorten the checking sequence.

Boute [2] shows how to generate a checking sequence for FSMs which may not have distinguishing sequences, but have distinguishing sets. The method also determines when transitions are "automatically" verified (i.e., when the verification of a transition is a consequence of the verification of other transitions) similarly to a more recent work [3].

All the above methods deal only with complete FSMs. Moreover, when the implementation under test has the reset feature, these methods do not attempt to use the reset input to shorten the checking sequence.

The contributions of this paper are twofold. First, we claim a theorem that states sufficient conditions for a sequence to be checking sequence for a (possibly partial) FSM with a distinguishing set. Notice that this theorem generalizes Ural *et al.*'s theorem [11]. The second contribution is a constructive method that generates checking sequence from FSM with a distinguishing set. Differently from recent works (namely, [11], [6], [3], [7], and [10]), the proposed method does not attempt to make a global optimization. Instead, it makes a local best choice in each step. Although the global optimization is expected to lead to shorter checking sequence, the graph-theoretical methods require that some choices be made *a priori* which may reduce the effectiveness of global optimization to that of local optimization if not lower. For instance, the method of Hierons and Ural [7] requires that a set of transfer sequences to be defined, and, moreover, the so-called α-set is determined by a separate algorithm, which may influence the effectiveness of the method. We present experimental results which indicate that the proposed method based only on local optimization performs better than existing methods in most cases.

This paper is organized as follows. In Section 2, we provide the necessary basic definitions. In Section 3, we define the notion of confirmed sets and use it to state sufficient conditions for an input sequence to be a checking sequence. A method for generating checking sequences based on the proposed conditions is presented in Sections 4. In Section 5, we present an experimental evaluation of the method. We discuss the related work in Section 6. Section 7 concludes the paper.

2 Definitions

A Finite State Machine is a deterministic Mealy machine, which is defined as follows.

Definition 1. *A* Finite State Machine (FSM) M *is a 6-tuple* $(S, s_0, I, O, D_M, \delta, \lambda)$, *where*

- S *is a finite set of states with the initial state* s_0,
- I *is a finite set of inputs,*
- O *is a finite set of outputs,*
- $D_M \subseteq S \times I$ *is a specification domain,*
- $\delta : D_M \to S$ *is a transition function, and*
- $\lambda : D_M \to O$ *is an output function.*

If $D_M = S \times I$, then M is a *complete* FSM; otherwise, it is a *partial* FSM. A tuple $(s, x) \in D_M$ is a (*defined*) *transition* of M. A string $\alpha = x_1 \ldots x_k$, $\alpha \in I^*$, is said to be a *defined* input sequence at state $s \in S$, if there exist $s_1, \ldots, s_{k+1}$, where $s_1 = s$, such that $(s_i, x_i) \in D_M$ and $\delta(s_i, x_i) = s_{i+1}$, for all $1 \leq i \leq k$. We use $\Omega(s)$ to denote the set of all defined input sequences for state s and Ω_M as a shorthand for $\Omega(s_0)$, i.e., for the input sequences defined for the initial state of M and, hence, for M itself.

Given sequences $\alpha, \varphi, \beta \in I^*$, if $\beta = \alpha\varphi$, then α is a *prefix* of β, denoted by $\alpha \leq \beta$, and φ is *suffix* of β. For a sequence $\beta \in I^*$, *pref*(β) is the set of prefixes of β, i.e., *pref*$(\beta) = \{\alpha \mid \alpha \leq \beta\}$. For a set of sequences T, *pref*(T) is the union of *pref*(β), for all $\beta \in T$.

We extend the transition and output functions from input symbols to defined input sequences, including the empty sequence ε, as usual, assuming $\delta(s, \varepsilon) = s$ and $\lambda(s, \varepsilon) = \varepsilon$, for $s \in S$, and for each $\alpha x \in \Omega(s)$, $\delta(s, \alpha x) = \delta(\delta(s, \alpha), x)$ and $\lambda(s, \alpha x) = \lambda(s, \alpha)\lambda(\delta(s, \alpha), x)$. Moreover, we extend the transition function to sets of defined input sequences. Given an FSM M, a set of input sequences $C \subseteq \Omega(s)$, $s \in S$, we define $\delta(s, C)$ to be the set of states reached by the sequences in C, i.e., $\delta(s, C) = \{\delta(s, \alpha) \mid \alpha \in C\}$. For simplicity, we slightly abuse the notation and write $\delta(s, C) = s'$, whenever $\delta(s, C) = \{s'\}$. Let also $\Phi(C, s) = \{\alpha \in C \mid \delta(s_0, \alpha) = s\}$, i.e., $\Phi(C, s)$ is the subset of sequences of C which lead M from the initial state to s, if any, thus containing the sequences of A converging on state s.

An FSM M is said to be *strongly connected*, if for each two states $s, s' \in S$, there exists an input sequence $\alpha \in \Omega(s)$, called a *transfer* sequence from state s to state s', such that $\delta(s, \alpha) = s'$.

Two states $s, s' \in S$ are *distinguishable*, denoted $s \nsim s'$, if there exists $\gamma \in \Omega(s) \cap \Omega(s')$, such that $\lambda(s, \gamma) \neq \lambda(s', \gamma)$. We also use the notation $s \nsim_\gamma s'$ when we need to refer to a sequence distinguishing states. If a sequence γ distinguishes each pair of distinct states, then γ is a *distinguishing* sequence. If γ distinguishes a state s from every other state, then γ is an *identification* sequence for state s. A distinguishing sequence is an identification sequence for each state, however, the converse does not hold. A *distinguishing set* $\boldsymbol{\Xi}$ is a set of $|S|$ identification sequences, such that for each pair of distinct states $s, s' \in S$, there exists a sequence distinguishing s and s' which is a common prefix of the respective identification sequences. Notice that, given a distinguishing sequence E, the set of the shortest prefixes of E, each of which is an

identification sequence of a state, is a distinguishing set. Moreover, for a given FSM, there may exist a distinguishing set even if no distinguishing sequence exists [2]. Given a set $C \subseteq \Omega(s) \cap \Omega(s')$, states s and s' are *C-equivalent*, denoted $s \sim_C s'$, if $\lambda(s, \gamma) = \lambda(s', \gamma)$ for all $\gamma \in C$. We define distinguishability and *C*-equivalence of machines as a corresponding relation between their initial states. An FSM M is said to be *reduced*, if all states are pairwise *distinguishable*, i.e., for all $s, s' \in S$, $s \neq s'$ implies $s \nsim s'$. An FSM N is *quasi-equivalent* to M, if $\Omega_M \subseteq \Omega_N$ and N is Ω_M-equivalent to M.

Given a reduced FSM M, let $\Im(M)$ be the set of all reduced complete FSMs with the input alphabet of M and at most n states, where n is the number of states of M.

Definition 2. *A finite input sequence $\omega \in \Omega_M$ of FSM M is a* checking sequence (for M), *if for each FSM $N \in \Im(M)$, such that $N \nsim M$, it holds that $N \nsim_\omega M$.*

Let $N = (Q, q_0, I, O', D_N, \Delta, \Lambda)$ be an arbitrary element of $\Im(M)$. Given an input sequence α, let $\Im_\alpha(M)$ be the set of all $N \in \Im(M)$, such that N and M are α-equivalent. Thus, ω is a checking sequence for M if every $N \in \Im_\omega(M)$ is quasi-equivalent to M.

A finite set $K \subseteq \Omega_M$ is a *state cover* for M if $\delta(s_0, K) = S$. A state cover K is *minimal* if $|K| = |S|$.

3 Generalizing Sufficient Conditions for Checking Sequences

Constructing checking sequence, a crucial issue is how to guarantee that the black box implementation is in a known state after the application of some input sequence. We propose a new way of addressing this issue by introducing the notion of confirmed sets of defined input sequences. In particular, a set of prefixes of an input sequence is confirmed if and only if it has transfer sequences for each state of the specification FSM M and any sequences that converge, i.e., lead to a same state (diverge, lead to different states) in any FSM that has the same output response to the given input sequence and has as many states as M if and only if they converge (diverge) in M.

Definition 3. *Let ω be a defined input sequence of an initially connected reduced FSM $M = (S, s_0, I, O, D_M, \delta, \lambda)$ and $K \subseteq pref(\omega)$. The set K is $\Im_\omega(M)$-*confirmed (*or simply* confirmed) *if K is a state cover and, for each $N \in \Im_\omega(M)$, it holds that for all $\alpha, \beta \in K$, $\Delta(q_0, \alpha) = \Delta(q_0, \beta)$ if and only if $\delta(s_0, \alpha) = \delta(s_0, \beta)$. An input sequence is* confirmed *if there exists a confirmed set that contains it.*

In this paper, we assume that the FSM M is strongly connected, reduced, and has a distinguishing set Ξ. Given Ξ, $\omega \in \Omega_M$ and $\alpha \leq \omega$, α is said to be Ξ-*recognized* in ω, if $\alpha E_s \leq \omega$,where $E_s \in \Xi$ and $\delta(s_0, \alpha) = s$.

Lemma 1. *Let $\omega \in \Omega_M$ and K be a minimal state cover. If each $\alpha \in K$ is Ξ-recognized in ω, then K is $\Im_\omega(M)$-confirmed.*

Proof. Let $N \in \Im_\omega(M)$ and $\alpha, \beta \in K$. We demonstrate that $\delta(s_0, \alpha) \neq \delta(s_0, \beta)$ implies $\Delta(q_0, \alpha) \neq \Delta(q_0, \beta)$. Suppose that $s = \delta(s_0, \alpha) \neq \delta(s_0, \beta) = s'$. Notice that α and β are followed in ω by E_s and $E_{s'}$, respectively. Thus, there exists a sequence $\gamma \in pref(E_s) \cap pref(E_{s'})$, such that $\lambda(s, \gamma) \neq \lambda(s', \gamma)$. As N is ω-equivalent to M, it follows that $\Lambda(\Delta(q_0, \alpha), \gamma) = \lambda(s, \gamma) \neq \lambda(s', \gamma) = \Lambda(\Delta(q_0, \beta), \gamma)$. Therefore, $\Delta(q_0, \alpha) \neq \Delta(q_0, \beta)$. ♦

From Lemma 1, we state the following corollary.

Corollary 1. *Let $\omega \in \Omega_M$ and K be a minimal state cover and $\Im_\omega(M)$-confirmed. Then K is a minimal state cover for any $N \in \Im_\omega(M)$.*

The next lemma indicates when a state cover K is confirmed, even if it is not minimal.

Lemma 2. *Let $\omega \in \Omega_M$ and K be a state cover. If each $\alpha \in K$ is Ξ-recognized in ω, then K is $\Im_\omega(M)$-confirmed.*

Proof. Let $N \in \Im_\omega(M)$. Let $\alpha, \beta \in K$. If $\delta(s_0, \alpha) \neq \delta(s_0, \beta)$, we can use the same argument used in the proof of Lemma 1 to prove that $\Delta(q_0, \alpha) \neq \Delta(q_0, \beta)$. Suppose then that $\delta(s_0, \alpha) = \delta(s_0, \beta)$. Let $K' \subseteq K$ be such that both $K_\alpha = K' \cup \{\alpha\}$ and $K_\beta = K' \cup \{\beta\}$ are minimal state covers for M. By Corollary 1, we have that K_α and K_β are also minimal state covers for N. Thus, $\Delta(q_0, \alpha)$ is distinct from each of the $n - 1$ states in $\Delta(q_0, K')$. As $\Delta(q_0, \beta)$ is also distinct from each of the $n - 1$ states in $\Delta(q_0, K')$ and N has n states, it follows that $\Delta(q_0, \alpha) = \Delta(q_0, \beta)$.♦

The next statement relies on the fact that if proper prefixes of some transfer sequences converge in a deterministic FSM, then the sequences converge as well. We use the following definitions. If α and $\alpha\varphi$ are confirmed sequences (in a confirmed set K), then φ is *verified* in $\delta(s_0, \alpha)$ (w.r.t. to K). If $x \in I$ is verified in s, then the transition (s, x) is *verified*.

Lemma 3. *Let K be a $\Im_\omega(M)$-confirmed set. If $\alpha, \beta \in K$, $\delta(s_0, \alpha) = \delta(s_0, \beta)$, and φ is verified in $\delta(s_0, \alpha)$, then the set $K \cup \{\beta\varphi\}$ is also $\Im_\omega(M)$-confirmed.*

Proof. As α and β are confirmed in K and φ is verified in $\delta(s_0, \alpha)$, $\alpha\varphi$ is also confirmed. Thus, we have that $\Delta(q_0, \alpha) = \Delta(q_0, \beta)$ and, therefore, it follows that $\Delta(q_0, \beta\varphi) = \Delta(\Delta(q_0, \beta), \varphi) = \Delta(\Delta(q_0, \alpha), \varphi) = \Delta(q_0, \alpha\varphi)$.♦

Thus, each sequence that is Ξ-recognized or is concatenation of Ξ-recognized and verified sequences can be included into a confirmed set. This key property of confirmed sets suggests a method for constructing a checking sequence which we elaborate later in the paper.

Theorem 1. *Let ω be an input sequence of a reduced FSM $M = (S, s_0, I, O, D_M, \delta, \lambda)$ with n states. ω is a checking sequence for M, if there exists a confirmed set K with the following properties*:

1. $\varepsilon \in K$.
2. *Each defined transition is verified.*

Proof. Let $N \in \Im_\omega(M)$. As M is strongly connected, for each $s \in S$, there exists $\alpha \in K$, such that $\delta(s_0, \alpha) = s$. For each $\beta \in K$, if $\delta(s_0, \beta) \neq \delta(s_0, \alpha)$, then $\Delta(q_0, \beta) \neq \Delta(q_0, \alpha)$. Thus, $|Q| = n$. Consequently, there exists a bijection $f : S \to Q$, such that for each $\alpha \in K$, $f(\delta(s_0, \alpha)) = \Delta(q_0, \alpha)$. As $\varepsilon \in K$, $f(s_0) = q_0$.

We prove that, for each $\nu \in \Omega_M$, $f(\delta(s_0, \nu)) = \Delta(q_0, \nu)$ using induction on ν, and, moreover, $\lambda(s, x) = \Lambda(f(s), x)$ for each $(s, x) \in D_M$.

If $\nu = \varepsilon$, we have $\nu \in K$, and, by definition, $f(\delta(s_0, \nu)) = \Delta(q_0, \nu)$. Let $\nu = \varphi x$ and assume that $f(\delta(s_0, \varphi)) = \Delta(q_0, \varphi)$. As the transition $(\delta(s_0, \varphi), x)$ is verified, there exist $\alpha, \alpha x \in K$, such that $\delta(s_0, \alpha) = \delta(s_0, \varphi)$. Thus, we have that $\Delta(q_0, \alpha) = f(\delta(s_0, \alpha)) = f(\delta(s_0, \varphi)) = \Delta(q_0, \varphi)$ and $f(\delta(s_0, \alpha x)) = \Delta(q_0, \alpha x)$. It follows that $f(\delta(s_0, \varphi x)) = f(\delta(\delta(s_0, \varphi), x)) = f(\delta(\delta(s_0, \alpha), x)) = f(\delta(s_0, \alpha x)) = \Delta(q_0, \alpha x) = \Delta(\Delta(q_0, \alpha), x) = \Delta(\Delta(q_0, \varphi), x) = \Delta(q_0, \varphi x)$. Therefore, $f(\delta(s_0, \varphi x)) = \Delta(q_0, \varphi x)$ and, by induction, for any $\nu \in \Omega_M$, $f(\delta(s_0, \nu)) = \Delta(q_0, \nu)$.

For each transition $(s, x) \in D_M$, there exists $\alpha x \in pref(\omega)$, $\delta(s_0, \alpha) = s$, $\alpha \in K$. Therefore, $\lambda(\delta(s_0, \alpha), x) = \Lambda(\Delta(q_0, \alpha), x)$. As $\alpha \in K$, we have that $\Delta(q_0, \alpha) = f(s)$ and, as N is ω-equivalent to M, it follows that $\lambda(s, x) = \Lambda(f(s), x)$.

Suppose finally that N can be distinguished from M. Therefore, there exists a sequence $\nu x \in \Omega_M$, such that $\lambda(s_0, \nu) = \Lambda(q_0, \nu)$ and $\lambda(s_0, \nu x) \neq \Delta(q_0, \nu x)$. There exist $\alpha \in K$, such that $\delta(s_0, \alpha) = \delta(s_0, \nu)$, and $\alpha x \in pref(T)$, such that $\lambda(\delta(s_0, \alpha), x) = \Lambda(f(\delta(s_0, \alpha)), x)$. $\delta(s_0, \alpha) = \delta(s_0, \nu)$ implies that $f(\delta(s_0, \alpha)) = f(\delta(s_0, \nu))$. Thus, $\lambda(\delta(s_0, \nu), x) = \Lambda(f(\delta(s_0, \nu)), x)$; and from $\lambda(s_0, \nu) = \Lambda(q_0, \nu)$, it follows that $\lambda(s_0, \nu x) = \Lambda(q_0, \nu x)$. The resulting contradiction concludes the proof. ♦

The theorem presented in [11] is a special case of Theorem 1. While Ural *et al.*'s theorem is applicable to complete FSMs with distinguishing sequence, Theorem 1 is applicable to partial FSMs as well as to FSMs with distinguishing sets. In the following section, we discuss how the sufficient conditions can be used to elaborate a method that generates checking sequence for FSMs with distinguishing sets. In Section 5, we present experimental results which demonstrate that the proposed method produces shorter checking sequences than known methods in most cases, even for complete FSMs with distinguishing sequences.

4 Checking Sequence Generation Method Based on Distinguishing Sets

In this section, we present a method based on distinguishing sets which exploits overlapping between identification sequences shortening the length of a checking sequence. The basic idea of the method is to consecutively append identification and transfer sequences to a current sequence ω, until a confirmed set $K \subseteq pref(\omega)$ is obtained and each transition of M is verified in K. We use $R(\omega)$ to denote the maximal subset of prefixes of ω, such that each $\alpha \in R(\omega)$ is either Ξ-recognized or there exist β, $\beta\varphi$, $\chi \in R(\omega)$, such that $\delta(s_0, \beta) = \delta(s_0, \chi)$ and $\alpha = \chi\varphi$ (Lemma 3). Notice that, if $R(\omega)$ is a state cover, by Lemma 1 and 2, $R(\omega)$ is a confirmed set. Notice also that if $\alpha \leq \omega$, then $R(\alpha) \subseteq R(\omega)$. Thus, the method obtains a checking sequence by guaranteeing that $R(\omega)$ is a confirmed state cover and that each transition is verified in $R(\omega)$. By $V(\omega)$ we denote the set of transitions verified in $R(\omega)$. Let also $U(\omega) = D_M \setminus V(\omega)$ be the set of unverified transitions.

The method is described in Algorithms 1 and 2 presented below. Let ω_i be a sequence obtained in the i-th iteration of Algorithm 1. There exist two cases that are dealt with by the algorithm. The first case occurs when $\omega_i \notin R(\omega_i)$. Then, we identify the longest suffix χ of ω_i, such that $\alpha_i\chi = \omega_i$ and χ is also a prefix of the identification sequence of $s = \delta(s_0, \alpha_i)$, and append E_s to α_i. Actually, as α_i is already followed by χ in ω_i, the suffix φ of E_s, with $\chi\varphi = E_s$, is appended to ω_i to obtain ω_{i+1}. Notice that $\alpha_i \in R(\omega_{i+1})$, since α_i is Ξ-recognized. After doing this a certain number of times, we have that $\omega_i \in R(\omega_i)$ (see Lemma 4 below), which is the second case. In this case, we verify a yet unverified transition (w.r.t. to $R(\omega_i)$). Notice that, as $\omega_i \in R(\omega_i)$, if α is a verified transfer sequence from $\delta(s_0, \omega_i)$ to some state s, we have that $\omega_i\alpha \in R(\omega_i\alpha)$. We then append $xE_{s'}$, for $s' = \delta(s, x)$, to $\omega_i\alpha$, so that $\omega_i\alpha x \in R(\omega_i\alpha xE_{s'})$, i.e., the

transition (s, x) is verified w.r.t. $R(\omega_i \alpha x E_{s'})$. Algorithm 1 terminates when no transition remains unverified. Notice that, the determination of $R(\omega_{i+1})$ and $U(\omega_{i+1})$ from a given intermediate sequence ω_i is a key feature of the algorithm. We provide an efficient method for doing this in Algorithm 2.

Algorithm 1
Input: A distinguishing set Ξ for a reduced FSM $M = (S, s_0, I, O, D_M, \delta, \lambda)$.
Output: A checking sequence ω
$i \leftarrow 0$
$\omega_0 \leftarrow \varepsilon$
$U(\omega_0) \leftarrow D_M$
$R(\omega_0) \leftarrow \varnothing$
while $U(\omega_i) \neq \varnothing$ **do**
Step 1. **if** $\omega_i \notin R(\omega_i)$, **then**
- let $\alpha_i \in pref(\omega_i)$ be the shortest prefix of ω_i, such that $\alpha_i \notin R(\omega_i)$, $\omega_i = \alpha_i \chi$, $s = \delta(s_0, \alpha_i)$, $E_s = \chi\varphi$.
- Update $\omega_{i+1} \leftarrow \omega_i \varphi$.
- Determine $R(\omega_{i+1})$ and $U(\omega_{i+1})$ using Algorithm 2 with the input ω_{i+1} and α_i.

Step 2. **else**,
- determine a shortest verified transfer sequence β_i from state $\delta(s_0, \omega_i)$ to some state s, such that there exists $x \in I$ and $(s, x) \in U(\omega_i)$.
- Let $\alpha_i = \omega_i \beta_i$ and $s' = \delta(s_0, \alpha_i x)$.
- Update $\omega_{i+1} \leftarrow \alpha_i x E_{s'}$
- Determine $R(\omega_{i+1})$ and $U(\omega_{i+1})$ using Algorithm 2 with the input ω_{i+1} and $\alpha_i x$.

end if
$i \leftarrow i + 1$
end while
Return $\omega \leftarrow \omega_i$

Now, we present an algorithm to calculate $R(\omega_{i+1})$ and $U(\omega_{i+1})$. Actually, these sets can be determined directly from their definitions. A straightforward method to find $R(\omega_{i+1})$ would require the inspection of all subsequences of ω_{i+1}. Notice, however, that it is sufficient to determine the set of verified sequences, since $R(\omega_{i+1})$ and $U(\omega_{i+1})$ can be derived from them. Suppose that the sequences β, $\beta\varphi$ and $\beta\varphi\chi$ are in $R(\omega_{i+1})$. Then, the sequences $\varphi\chi$ and φ are verified in $\delta(s_0, \beta)$ and χ is verified in $\delta(s_0, \beta\varphi)$. The fact that $\varphi\chi$ is verified in $\delta(s_0, \beta)$ is not used to determine $R(\omega_{i+1})$ and $U(\omega_{i+1})$, since the same result is obtained from the fact that the other two sequences are verified. This observation suggests that only shortest verified sequences have to be considered for a given state. We denote by $P(\omega_{i+1})$ the maximal subset of $S \times (I^* \backslash \{\varepsilon\})$, such that $(s, \alpha) \in P(\omega_{i+1})$ iff α is the shortest sequence verified in s. Thus, to determine $R(\omega_{i+1})$ and $U(\omega_{i+1})$, we first determine $P(\omega_{i+1})$ as follows. Notice that $P(\omega_0) = \varnothing$. We identify the

longest $\beta \in R(\omega_{i+1})$, such that $\alpha = \beta\varphi$, for some non-empty φ, and include the pair $(\delta(s_0, \beta), \varphi)$ in $P(\omega_{i+1})$. Notice that if α is the only recognized sequence, i.e., if $R(\omega_i) = \emptyset$, such a sequence β does not exist, in which case $P(\omega_{i+1})$ is empty. After the inclusion of a new pair into $P(\omega_{i+1})$, we check whether some sequences can be removed from $P(\omega_{i+1})$, so that it contains only the shortest verified sequences.

Once $P(\omega_{i+1})$ is determined, we can obtain $R(\omega_{i+1})$ and $U(\omega_{i+1})$ as follows. If α is recognized, for each $(\delta(s_0, \alpha), \varphi) \in P(\omega_{i+1})$, we include $\alpha\varphi \in pref(T)$ in $R(\omega_{i+1})$. The set of verified transitions $V(\omega_{i+1})$ is now $D_M \cap P(\omega_{i+1})$ and, consequently, $U(\omega_{i+1})$ becomes $D_M \setminus V(\omega_{i+1})$. The following algorithm shows how $P(\omega_{i+1})$ is determined after the Ξ-recognition of a sequence α and how $R(\omega_{i+1})$ and $U(\omega_{i+1})$ are obtained from $P(\omega_{i+1})$.

Algorithm 2

Input: Sequence ω_{i+1} and Ξ-recognized sequence α; the sets $R(\omega_i)$ and $U(\omega_i)$.

Output: $R(\omega_{i+1})$ and $U(\omega_{i+1})$

- $R(\omega_{i+1}) \leftarrow R(\omega_i) \cup \{\alpha\}$
- $P(\omega_{i+1}) \leftarrow P(\omega_i)$
- **if** $R(\omega_i) \neq \emptyset$ **then**
    - Let β be the longest sequence in $R(\omega_{i+1})$. Let $s = \delta(s_0, \beta)$ and φ be the such that $\beta\varphi = \alpha$.
    - $P(\omega_{i+1}) \leftarrow P(\omega_{i+1}) \cup \{(s, \varphi)\}$
    - **while** there exist $(s, \tau), (s, \chi) \in P(\omega)$, such that $\tau = \chi\gamma$ and $\gamma \neq \varepsilon$ **do**
      $P(\omega_{i+1}) \leftarrow P(\omega_{i+1}) \setminus \{(s, \tau)\} \cup \{(\delta(s, \chi), \gamma)\}$
    - **end while**
    - $V(\omega_{i+1}) \leftarrow D_M \cap P(\omega_{i+1})$
    - $U(\omega_{i+1}) \leftarrow D_M \setminus V(\omega_{i+1})$
    - **for each** $\chi \in pref(\omega_{i+1}) \setminus R(\omega_{i+1})$, $s = \delta(s_0, \chi)$ **do**
      for each $(s, \varphi) \in P(\omega_{i+1})$ **do**
      $R(\omega_{i+1}) \leftarrow R(\omega_{i+1}) \cup (\{\chi\varphi\} \cap pref(\omega_{i+1}))$
      end for
    - **end for**
- **end if**

Return $R(\omega_{i+1})$ and $U(\omega_{i+1})$

The next lemma states that Step 1 of Algorithm 1 cannot be executed infinitely many times. This lemma is important to prove that Algorithm 1 terminates and that the obtained sequence is actually a checking sequence.

Lemma 4. *In Algorithm 1, Step 1 can be executed at most* $\sum_{s \in S} |E_s|$ *times without executing Step 2.*

Proof. We show that, for a given $s \in S$, the number of executions of Step 1 when α_i is such that $\delta(s_0, \alpha_i) = s$ is at most $|E_s|$. Let α_i and α_j be the shortest Ξ-recognized sequences (obtained in the i-th and the j-th iterations of Algorithm 1, respectively), such that $i < j$ and $\delta(s_0, \alpha_i) = \delta(s_0, \alpha_j) = s$. Notice that $\alpha_i \notin R(\omega_i)$, but $\alpha_i \in R(\omega_{i+1})$. If

$\omega_{i+1} \in R(\omega_{i+1})$, then Step 2 must be executed. Therefore, suppose that $\omega_{i+1} \notin R(\omega_{i+1})$. In the next iteration of the algorithm, we have that $\omega_{i+1} = \alpha_i E_s$ and, thus, $\alpha_{i+1} < \alpha_i E_s$. Let β_i be the non-empty prefix of E_s, such that $\alpha_{i+1} = \alpha_i \beta_i$. Consider now α_j, i.e., the sequence which is Ξ-recognized in the j-th iteration of the algorithm. It follows from the definition of R that $\alpha_j \beta_i \in R(\omega_{j+1})$, since $\delta(s_0, \alpha_i) = \delta(s_0, \alpha_j)$ and α_i, α_j, $\alpha_i \beta_i \in R(\omega_{j+1})$. In the next iteration, we have that $\omega_{j+1} = \alpha_j E_s$ and, thus, $\alpha_{j+1} \leq \alpha_j E_s$. Then, there must exist a non-empty sequence $\beta_j \leq E_s$, such that $\alpha_{j+1} = \alpha_j \beta_j$ and $\beta_i < \beta_j \leq E_s$. As $|\beta_i| < |\beta_j| \leq |E_s|$, it follows that there exist at most $|E_s|$ executions of Step 1, such that $\delta(s_0, \alpha) = s$. ♦

Theorem 3. *Let ω be a sequence obtained by Algorithm 1. Then, ω is a checking sequence.*

Proof. When the algorithm terminates, $U(\omega) = \varnothing$, which implies that each transition is verified in $R(\omega)$. The algorithm indeed terminates because after each execution of Step 2, the number of unverified transitions is decreased by at least one. Therefore, Step 2 can be executed at most $|D_M|$ times. As, by Lemma 4, after a finite number of executions of Step 1, Step 2 must be executed, the number of iterations of the algorithm is finite.

In the first iteration of the algorithm, $\omega_0 = \varepsilon$ and $R(\omega_0) = \varnothing$. Then, Step 1 is executed and yields $\omega_1 = E_{s0}$. Thus, $\varepsilon \in R(\omega_1)$ and, consequently, $\varepsilon \in R(\omega)$.

We now show that $R(\omega)$ is a $\Im_\omega(M)$-confirmed set. By the definition of $R(\omega)$, we have that each $\alpha \in R(\omega)$ is either (i) Ξ-recognized or (ii) there exist β, $\beta\varphi$, $\chi \in R(\omega)$, such that $\delta(s_0, \beta) = \delta(s_0, \chi)$ and $\alpha = \chi\varphi$. Let $K \subseteq R(\omega)$ be the set of Ξ-recognized sequences. Observe first that, in the case (ii), we have that $\delta(s_0, \beta\varphi) = \delta(s_0, \alpha)$, which implies that if $\alpha \in R(\omega)$ is not Ξ-recognized, then another sequence that takes M to the same state as α should be. Thus, if there exists a sequence $\alpha \in R(\omega)$, $\delta(s_0, \alpha) = s$, there must exist at least one sequence $\beta \in \Phi(K, s)$. As each transition is verified and M is strongly connected, for each state s, there exists a sequence $\alpha \in \Phi(R(\omega), s)$. Thus, there exists $\beta \in \Phi(K, s)$ for each state s. Consequently, K is a state cover. By Lemma 2, K is a confirmed set. By the definition of $R(\omega)$ and Lemma 3, it follows that $R(\omega)$ is a confirmed set and satisfies the conditions of Theorem 1. Thus, ω is a checking sequence. ♦

4.1 An Example

We now illustrate the application of the method to the FSM M_1 in Figure 1. This machine has a distinguishing sequence $E = aa$. The distinguishing set contains three identification sequences $E_1 = E_2 = aa$, and $E_3 = a$. The intermediate values of ω_i, $R(\omega_i)$, and $U(\omega_i)$ are presented in Table 1. The obtained checking sequence ω = *aaaaababaabaa* has length of 13. For the same FSM, the method in [6] finds a checking sequence of length 32. An improved version of the method generates a checking sequence of length 15 [10].

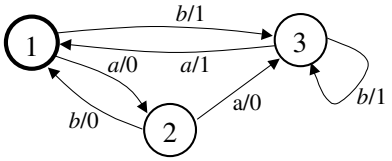


Fig 1. Complete FSM M_1 from [11].

Table 1. Execution of Algorithm 1

i	Step Executed	ω_i	Recognized Prefixes $R(\omega_i)$	Unverified Transitions $U(\omega_i)$
0		ε	∅	{(1, *a*), (1, *b*), (2, *a*), (2, *b*), (3, *a*), (3, *b*)}
1	Step 1	*aa*	{ε}	{(1, *a*), (1, *b*), (2, *a*), (2, *b*), (3, *a*), (3, *b*)}
2	Step 1	*aaa*	{ε, *a*, *aa*}	{(1, *b*), (2, *b*), (3, *a*), (3, *b*)}
3	Step 1	*aaaaa*	{ε, *a*, *aa*, *aaa*, *aaaa*, *aaaaa*}	{(1, *b*), (2, *b*), (3, *b*)}
4	Step 2	*aaaaaba*	{ε, *a*, *aa*, *aaa*, *aaaa*, *aaaaa*, *aaaaab*, *aaaaaba*}	{(1, *b*), (2, *b*)}
5	Step 2	*aaaaababaa*	{ε, *a*, *aa*, *aaa*, *aaaa*, *aaaaa*, *aaaaab*, *aaaaaba*, *aaaaabab*, *aaaaababaa*}	{(2, *b*)}
6	Step 2	*aaaaababaabaa*	{ε, *a*, *aa*, *aaa*, *aaaa*, *aaaaa*, *aaaaab*, *aaaaaba*, *aaaaabab*, *aaaaababaa*, *aaaaababaab*, *aaaaababaaba*, *aaaaababaabaa*}	∅

4.2 Reset Feature

An FSM M has a reset feature if it has a special input (denoted r), which transfers it, as well as all its possible implementation machines, from any state to the initial state producing a null output, usually represented by the empty sequence ε. The reset transitions are assumed to be correct, i.e., verified; so Theorem 1 directly applies to FSMs with reset feature. Moreover, Algorithms 1 and 2 can be extended to deal with cases where the reset input is available. By default, the reset transitions are verified in each state and can be used to construct verified transfer sequences, which may result in shorter checking sequences. In Step 2 of Algorithm 1, when searching for a shortest verified transfer sequence β_i, the reset input may be used. In Algorithm 2, we have that $(s, r) \in P(\omega)$, for each state s and any input sequence ω.

Consider the partial FSM M_2 in Figure 2 (dashed lines represent the reset transitions) and the distinguishing set Ξ with $E_1 = E_2 = E_3 = E_4 = aa$. If Algorithm 1 is applied to this FSM, it generates the checking sequence ω = *aaaaabaaabaabbabaa* with length 18. However, if M_2 has the reset feature, the algorithm generates the checking sequence ω_r = *aaaaabaaabaarabaa* with length 17. The execution of the algorithm either with or without reset is the same up to the point where ω_i = *aaaaabaaabaa*. At that point, the only unverified transition remains (2, *b*). Notice that $\delta(s_0, \omega_i) = 3$. The shortest verified transfer sequence from state 3 to state 2 is *ra*, if the reset input is available, or *bba*, otherwise. In this case, the reset input contributes to shortening checking sequence.

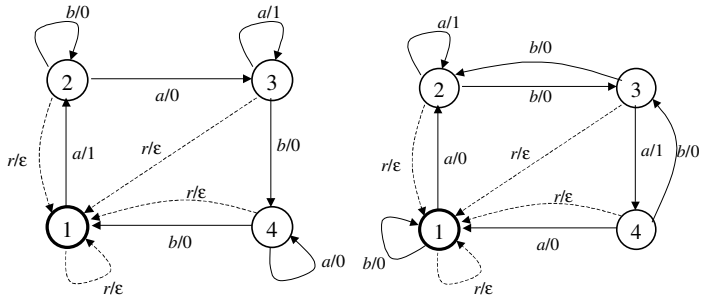


Fig. 2. (a) Partial FSM M_2 with reset feature; (b) Complete FSM M_3 with reset feature.

There are cases, however, where the reset feature increases the length of a checking sequence generated by Algorithm 1, since the best local choices (e.g., shortest paths) made do not guarantee to lead to globally optimized checking sequences. Consider, for instance, M_3 in Figure 2 with the distinguishing sequence *aa*. The checking sequence without reset is ω = *aaaabaaaabbaababaabaa* of length 21, while the checking sequence with reset is ω_r = *aaaabaaaabbaarbaababaa* of length 22. The execution of the algorithm in both cases is the same, up to the point where ω_i = *aaaabaaaabbaa*, $\delta(s_0, \omega_i) = 2$. At this point, $U(\omega_i) = \{(1, b), (4, b)\}$. If reset is not available, the shortest path chosen in Step 2 of Algorithm 1 is β_i = *ba*, which transfers to state 4. After this step, we have that ω_{i+1} = *aaaabaaaabbaababaa*, $\delta(s_0, \omega_{i+1}) = 1$, and $U(\omega_{i+1}) = \{(1, b)\}$. Then, Step 2 is executed again, choosing $\beta_{i+1} = \varepsilon$. On the other hand, if reset is used, the shortest path chosen in Step 2 is β_i = *r*, which transfers to state 1. After this step, we have that ω_{i+1} = *aaaabaaaabbaarbaa*, $\delta(s_0, \omega_{i+1}) = 2$, and $U(\omega_{i+1}) = \{(4, b)\}$. Then, Step 2 is executed, choosing β_{i+1} = *ba*. An interesting question investigated in Section 5.1 is the impact of using the reliable reset feature on the length of checking sequence for randomly generated FSMs.

5 Experimental Results

This section describes an experimental evaluation of the method for constructing checking sequence proposed in this paper and some existing methods. We also investigate the reduction provided by the use of the reset feature. The experiments involve randomly generated FSMs. Since the existing methods treat only complete FSMs with distinguishing sequences, only such machines are considered to have a fair comparison.

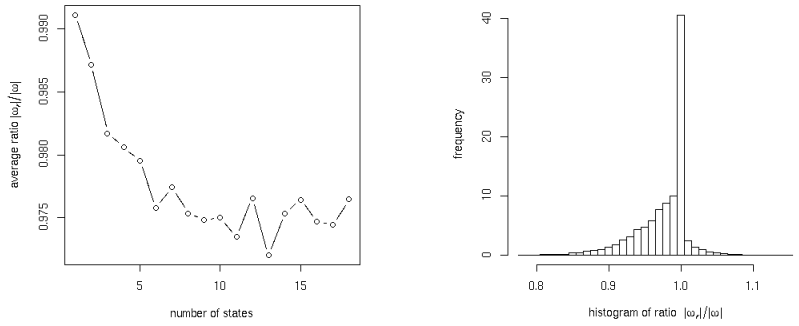
We generate complete strongly connected reduced FSMs with a distinguishing sequence in the following way. Sets of states, inputs, and outputs with the required number of elements are first created. The generation proceeds then in three phases. In the first phase, a state is selected as the initial state and marked as "reached". Then, for each state *s* not marked as "reached", the generator randomly selects a reached state *s'*, an input *x*, and an output *y* and adds a transition from *s'* to *s* with input *x* and

output *y*, and mark *s* as "reached". When this phase is completed, an initially connected FSM is obtained. In the second phase, the generator adds transitions (by randomly selecting two states, an input, and an output) to the machine until a complete FSM is obtained. If the FSM is not strongly connected, it is discarded and another FSM is generated. In the third phase, a distinguishing sequence is searched. If the FSM does not have a distinguishing sequence, it is discarded and another FSM is generated.

5.1 Reset Feature

As discussed in Section 4, our method can be applied to FSMs with the reset feature. The use of the reset input can result in shorter transfer sequences, which may shorten the resulting checking sequence. In this experiment, we evaluate the reduction obtained by the usage of the reset input. We randomly generated FSMs which have distinguishing sequence. Each FSM has two inputs, two outputs, and the number of states *n* ranging from three to 20. For each value of *n*, 1000 FSMs are generated.

For each FSM, a checking sequence ω is obtained using the proposed method. Then, we augmented the machine with the reset input and executed the method on the resulting FSM, obtaining a checking sequence ω_r. Figure 3(a) characterizes the variation of the average ratio $|\omega_r|/|\omega|$ with respect to the number of states. We observe that on average ω_r is about 2.5% shorter than ω. Figure 3(b) shows the frequency of the obtained reduction. Notice that in about 40% of the cases, the ratio is between 0.995 and 1.005, indicating that the reset feature has a little impact on the length of the checking sequence (at least for the chosen FSM parameters). However, in some cases, the checking sequence for the FSM augmented with the reset input is 20% shorter. On the other hand, in some cases it may be 10% longer. Thus, our experiments indicate that the reset feature does not significantly influence the length of checking sequences. However, more large scale experiments may be needed to confirm this conclusion.

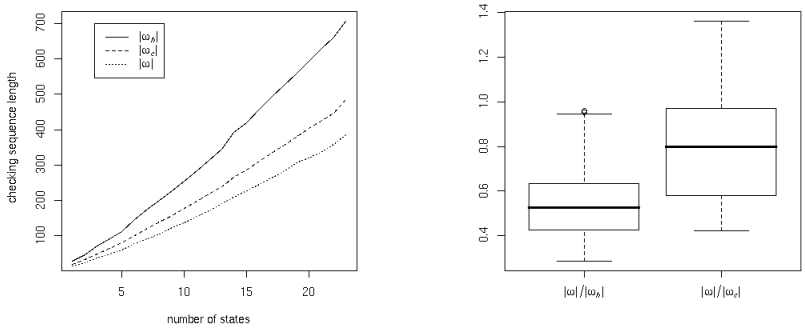


(a) (b)

Fig. 3: Reduction ratio of the length of checking sequences with and without the reset input: (a) average ratio variation with respect to the number of states; (b) histogram of the ratio.

5.2 Comparison with Existing Methods

In this experiment, we compare the length of checking sequences generated by the proposed method and by the methods presented in [7] and [3]. We chose to consider machines with two inputs, two outputs, and number of states n ranging from three to 25. For each value of n, we randomly generated 1000 FSMs which have distinguishing sequence. For each FSM, we executed Algorithm 1, generating checking sequence ω. No execution took more than one second. Then, we executed Hierons and Ural's method [7], obtaining checking sequence ω_h, and Chen *et al.*'s method [3], which results in sequence ω_c. Figure 4(a) shows the average length of ω, ω_h and ω_c. The experimental data indicate that ω is, on average, 45% shorter than ω_h, while ω is, on average, 20% shorter than ω_c.



(a) (b)

Fig. 4: (a) Average lengths of ω, ω_h and ω_c; (b) Boxplots of the ratios $|\omega|/|\omega_h|$ and $|\omega|/|\omega_c|$.

Figure 4(b) shows the boxplots of the ratios $|\omega|/|\omega_h|$ and $|\omega|/|\omega_c|$. Notice that the maximum value of $|\omega|/|\omega_h|$ is smaller than 1.0, thus, the proposed method generated shorter checking sequences than the Hierons and Ural's method in all experiments. Notice also that the reduction may be as high as 70%. On the other hand, compared with the Chen *et al.*'s method, method proposed in this paper generated shorter checking sequences in 75% of the cases, since the 3rd quartile of the boxplot is below 1.0. However, the maximum value of $|\omega|/|\omega_h|$ is 1.35, as in some experiments, the proposed method generated sequences 35% longer than Chen *et al.*'s method. On the other hand, in some cases our method generated sequences 60% shorter than Chen *et al.*'s method.

6 Related Work

There has been much interest in the generation of checking sequence, pioneered by Hennie's work [5]. Hennie discusses an approach for designing a checking sequence based on a distinguishing sequence. The method consists of two parts. The states are

assumed to be ordered. In the first part, the distinguishing sequence is applied to each state, starting from the initial state, and transfer sequences are used to lead FSM to the second state. In the second part, each transition (s_i, x) is checked. To do this, the FSM is brought to a known state. This is done by transferring FSM to s_{i-1}, applying the distinguishing sequence, transferring to s_i (the same sequence used in the first part must be used), and applying x followed by the distinguishing sequence. Notice that Hennie does not assume that the FSM is initialized, thus checking sequence generated by his method should be prepended by a synchronizing or homing sequence, in order to bring both the specification and the implementation to a known state. Differently, we assume that the specification and the implementation are initialized and, thus, we do not use a synchronizing or homing sequence.

Gonenc [4] presents an algorithmic approach to generate checking sequence for FSMs with distinguishing sequences. The method, known as the method D, is divided into two parts, similarly to Hennie's work. In the first part, an α-sequence is generated, such that the distinguishing sequence is guaranteed to distinguish the states in the implementation. In an α-sequence, the distinguishing sequence is applied to each state of the FSM, using transfer sequences, if necessary. In the second part, a β-sequence is generated, such that each transition is checked. A β-sequence is a concatenation of transitions, followed by the distinguishing sequence, using transfer sequences, if necessary. Notice that α- and β-sequences correspond to the sequences generated in the first and the second part of Hennie's method, respectively. The α- and β-sequences are, then, concatenated to form a checking sequence. The number of transitions that are checked is reduced, using the fact that the last transition of the distinguishing sequence when applied to a state will be checked when all the other transitions of the distinguishing sequence were checked.

Kohavi and Kohavi [8] show that, instead of whole distinguishing sequence, suitable prefixes of it can be used to shorten the checking sequence. Boute [2] further shows that shorter sequences can be obtained if, instead of distinguishing sequences, distinguishing sets are used, and if the overlapping among the identification sequences is exploited.

Ural *et al.* [11] propose a method that attempts to minimize the length of checking sequence generated using distinguishing sequence. Sufficient conditions for a sequence to be a checking sequence are formulated there and used in several work, e.g., [6], [3], [10], [7], and [12]. The conditions are now further relaxed in Theorem 1 of this paper. The α- and β-sequences of Gonenc's method are divided in [11] into smaller pieces (the α-set and the set of transition tests) that are combined with appropriate transfer sequences to form a checking sequence. The problem of finding a minimal length checking sequence is then cast as a Rural Chinese Postman Problem (RCPP), as previously proposed by Aho *et al.* [1]. The RCPP is an NP-complete problem of finding a minimal tour which traverses some required edges in an appropriate graph. Ural *et al*. show how a graph can be defined, such that the RCPP tour satisfies the stated sufficient conditions and, thus, it is a checking sequence. An α-set and a set of transfer sequences must be provided as input parameters of the method. Improvements to this method are proposed by Hierons and Ural [6] [7].

Chen *et al.* [3] demonstrate that the verification of last transition traversed by a distinguishing sequence applied to a particular state is implied by the verification of

the other transitions. The authors present an (NP-complete) algorithm that identifies transitions to remove sequences ensuring their verification from the RCPP graph. The possibility of overlapping the distinguishing sequence, as proposed by Boute [2], is exploited by Ural and Zhang [10]. The RCPP graph modelling of Hierons and Ural [6] is modified, so that edges with negative cost are added to represent the possible overlapping. However, incorporating sequence overlapping comes with the price, since the size of the RCPP grows significantly.

All the above methods only deal with complete FSMs (however, partial machines can also be allowed, provided that definitions are adjusted as in this paper), and do not attempt to use the reset input to shorten the checking sequence. On the other hand, the method proposed in this paper can be used for generating checking sequence for a possibly partial FSM with a distinguishing set and, possibly, with the reset feature. The proposed method makes a local best choice in each step. This approach diverges from methods proposed in recent work (namely, [11], [6], [3], [7], [10]), which use graph-theoretical modeling for minimizing the length of checking sequence. We consider that our local optimization based approach has at least two advantages over the graph-theoretical ones, discussed below.

Firstly, the graph-theoretical methods attempt to globally optimize the length of checking sequence, but only after some input parameters are set, e.g., the α-sequences and transfer sequence set used by Ural *et al.* [11]. However, the length of checking sequence is influenced by these parameters and, thus, a sub-optimized sequence may be generated anyway. On the other hand, the method proposed in this paper does not require any input, besides the distinguishing set. Instead of assuming that suitable parameters are furnished, the algorithm makes choices based on the information available up to a certain execution point. The results of an experimental comparison indicate that the proposed method produces shorter checking sequences than existing methods in most cases. However, more experiments are needed to order to find a proper compromise between global and local optimization in generating checking sequence.

The second advantage of our approach is in its extensibility. For instance, it is not immediately clear how to the ideas of [10], [3] and [12] can be integrated in a same method, since each requires adaptations of the graph model which are not straightforward to merge. Our approach is based on a new problem casting, using the notion of "confirmed sequences". This formulation allows us to relax the existing sufficient conditions and generalize them to partial reduced FSM with distinguishing sets. Further generalizations constitute our current work.

7 Conclusion

In this paper, we stated sufficient conditions for an input sequence to be a checking sequence for possibly partial FSMs with distinguishing sets. Based on these conditions, we proposed a method for generating checking sequence. The method can be used either with or without the reset feature. Moreover, the method allows the use of distinguishing sets, while recent methods deal only with FSMs with distinguishing sequences.

We experimentally compared the proposed method with existing generation methods, using randomly generated complete FSMs with distinguishing sequences. The results indicate that the proposed method generates shorter checking sequence in most cases. We also presented an experimental evaluation of the impact of the reset feature on the length of checking sequence. We noticed that, although shorter sequences may sometimes be obtained, our preliminary experiments indicate that the reset feature does not significantly influence the length of checking sequences.

As future work, we can mention several possible extensions of the presented results. For instance, the improvement suggested by Chen *et al.* [3] can be incorporated into the method proposed in this paper. The set of unverified transitions may be initialized with a subset of the defined transitions, following the algorithm of Chen *et al*. Finally, our experiments show that some checking sequences do not satisfy the suggested sufficient conditions, so there is still room for improvements on the conditions, which may lead to shorter checking sequences.

References

1. Aho, A.V., Dahbura, A.T., Lee, D., Uyar, M.U.: An optimization technique for protocol conformance test generation based on UIO sequences and rural chinese postman tours. IEEE Transactions on Communications 39(11) (1991) 1604–1615
2. Boute, R.T.: Distinguishing sets for optimal state identification in checking experiments. **23**(8) (1974) 874–877
3. Chen, J., Hierons, R.M., Ural, H., Yenigun, H.: Eliminating redundant tests in a checking sequence. In: TestCom 2005. Number 3502 in LNCS (2005) 146–158
4. Gonenc, G.: A method for the design of fault detection experiments. IEEE Transactions on Computers **19**(6) (1970) 551–558
5. Hennie, F.C.: Fault-detecting experiments for sequential circuits. In: Proceedings of Fifth Annual Symposium on Circuit Theory and Logical Design. (1965) 95–110
6. Hierons, R.M., Ural, H.: Reduced length checking sequences. IEEE Transactions on Computers **51**(9) (2002) 1111–1117
7. Hierons, R.M., Ural, H.: Optimizing the length of checking sequences. IEEE Transactions on Computers **55**(5) (2006) 618–629
8. Kohavi, I., Kohavi, Z.: Variable-length distinguishing sequences and their application to the design of fault-detection experiments. IEEE Transactions on Computers **17**(8) (1968) 792–795
9. Moore, E.F.: Gedanken-experiments on sequential machines. Automata Studies, Annals of Mathematical Studies (34) (1956) 129–153
10. Ural, H., Zhang, F.: Reducing the lengths of checking sequences by overlapping. In: TestCom 2006. Number 3964 in LNCS (2006) 274–288
11. Ural, H., Wu, X., Zhang, F.: On minimizing the lengths of checking sequences. IEEE Transactions on Computers **46**(1) (1997) 93–99
12. Yalcin, M.C., Yenigun, H.: Using distinguishing and uio sequences together in a checking sequence. In: TestCom 2006. Number 3964 in LNCS (2006) 274–288