Checking Sequence Generation Using State Distinguishing Subsequences

Adenilso Simão [+, *]
[+]*Instituto de Ciências Matemáticas e de Computação*
Universidade de São Paulo
São Carlos, São Paulo, Brazil
adenilso@icmc.usp.br

Alexandre Petrenko
[*]*Centre de recherche informatique de Montreal (CRIM)*
Montreal, Quebec, Canada, petrenko@crim.ca

Abstract—**A checking sequence generated from a Finite State Machine (FSM) is used in testing to demonstrate correctness of an implementation under test. It can be obtained by concatenating inputs triggering state transitions followed by final state verification sequences. Usually, the latter are derived from a distinguishing set or sequence, assuming that a given FSM possesses it. It has been suggested that, under certain conditions, Unique Input/Output (UIO) sequences can also be used. In this paper, we propose using sequences with less state distinguishability power than distinguishing and UIO sequences. Such sequences are shorter and thus can reduce the length of checking sequences. We formulate conditions under which such sequences can replace distinguishing and UIO sequences and elaborate a checking sequence generation method based on these conditions. An example is provided to demonstrate that the proposed method yields a checking sequence shorter than existing methods.**

Keywords—checking sequence, FSM testing, test generation methods

I. INTRODUCTION

Test generation from a Finite State Machine (FSM) has received significant attention over the last decades. Given a specification FSM and a black box implementation, the objective is to construct a test suite that can check whether the implementation behaves correctly according to the specification FSM. Usually, the implementation is assumed to behave like an unknown FSM with the same input alphabet and at most as many states as the specification FSM. 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], [10], [4], [2], [16], [6], [3], [15], [7], [17], and [14]. These methods use various techniques to guarantee that a black box implementation is in a known state after the application of some input sequence. Then, the transitions of the FSM are verified, by transferring it to a starting state, applying the input of the transition, and guaranteeing that the reached state is correct. The task of identifying a state the implementation reaches after the application of an input sequence is simplified if the specification FSM M possesses a so-called distinguishing input sequence which produces a distinct output sequence for each state of M. However, not every FSM has a distinguishing sequence. A distinguishing set is a set of input sequences, one for each state of M, (often called a UIO sequence) such that for each pair of distinct states, the respective input sequences have a common prefix for which both states produce different outputs. Notice that, while the sequences in a distinguishing set are always UIO sequences, a set of UIO sequences may not form a distinguishing set. A distinguishing set can also be determined from a distinguishing sequence. However, there exist FSMs with a distinguishing set which do not have distinguishing sequence [2], [14]. It is interesting to note that distinguishing sets correspond to adaptive distinguishing sequences investigated in [11].

Several generation methods have been proposed for generating checking sequence when a distinguishing sequence is available, e.g., [5], [4], [16], [6], [3], [15], [7], and [17]. Ural *et al.* [16] state an important theorem with sufficient conditions for a sequence to be a checking sequence for a complete FSM with a distinguishing sequence. A generation method is also suggested, which has later been improved in [6] and [7]; Chen *et al.* [3] demonstrate that it is sufficient to consider only a subset of the FSM transitions; and Ural and Zhang [15] use the overlapping of the distinguishing sequence with itself to shorten the checking sequence.

In [14] we stated sufficient conditions for a sequence to be checking sequence for an FSM with a distinguishing set, generalizing Ural *et al.*'s theorem [16]. Then, we elaborated a checking sequence generation method which is applicable to partial FSMs. In the same vein as the work of [2] and [11], and differently from most of recent work, the proposed method requires that the FSM has a distinguishing set (also known as an adaptive distinguishing sequence). Moreover, differently from other work (namely, [16], [6], [3], [7], and [15]), the proposed method makes a local best choice in

each step instead of solving a global optimization problem, thus scaling better, as experimental results indicate [14].

It has been recently demonstrated [17] that UIO sequences can be used in conjunction with distinguishing sequence while building checking sequence. UIO sequences are usually shorter than distinguishing sequence and may thus lead to shorter checking sequences.

In this paper, we investigate a new avenue for shortening checking sequences, namely, the use of sequences with less state distinguishability power than distinguishing and UIO sequences. Such sequences are usually shorter and can reduce the length of checking sequences. We formulate conditions under which such sequences can replace distinguishing and UIO sequences. The conditions are used to elaborate a checking sequence generation method.

The rest of the paper is organized as follows. In Section 2, we provide the necessary basic definitions. In Section 3, we investigate conditions for using sequences with lesser state distinguishability power than distinguishing and UIO sequences while checking sequence is constructed. In Section 4, we develop a checking sequence generation method based on the proposed conditions. In Section 5, the method is illustrated using an example which demonstrates that it can produce shorter checking sequences than existing methods. We compare the contributions of this paper with the related work in Section 6. Section 7 concludes the paper.

II. Basic 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 $1 \le i \le k$. 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.

Figure 1 shows an example of a complete FSM, used in [7] and [3]. An edge from state *s* to state *s′*, with label *x/y* indicates that (s, x) is a defined transition, $\delta(s, x) = s'$, and $\lambda(s, x) = y$. For instance, $\delta(5, b) = 3$ and $\lambda(5, b) = 1$.
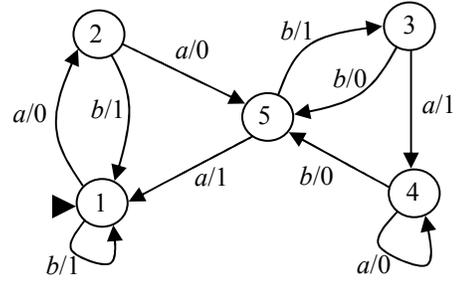

Figure 1: Complete FSM.

Given sequences $\alpha, \beta, \gamma \in I^*$, if $\beta = \alpha\gamma$, then α is a *prefix* of β, denoted by $\alpha \le \beta$, and γ is a *suffix* of β. We also say that a prefix of γ *extends* α (*in* β) and that β is an *extension* of α. For a sequence $\beta \in I^*$, *pref*(β) is the set of prefixes of β, i.e., *pref*(β) = $\{\alpha \mid \alpha \le \beta\}$. 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 $\delta(s, C) = \{s'\}$.

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 $\delta(s, \alpha) = s'$. The FSM *M* is *initially connected* if there is a transfer sequence from the initial state s_0 to each state.

Given a set $C \subseteq \Omega(s) \cap \Omega(s')$, states *s* and *s′* are *C-equivalent*, 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*. An FSM *N* is *quasi-equivalent* to *M*, if $\Omega_M \subseteq \Omega_N$ and *N* is Ω_M-equivalent to *M*.

Two states $s, s' \in S$ are *distinguishable*, if there exists $\gamma \in \Omega(s) \cap \Omega(s')$, such that $\lambda(s, \gamma) \ne \lambda(s', \gamma)$. It is said that γ distinguishes *s* and *s′*. 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*, also known as Unique Input/Output (UIO) sequence for state *s*. A distinguishing sequence is an identification sequence for each state, however, the converse does not hold. Boute [2] introduces the notion of distinguishing sets. A *distinguishing set* Ξ 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. A compound distinguishing sequence is a set of input sequences H having a common prefix χ, such that for each set P of $\{\chi\}$-equivalent states, there exists a sequence in H that distinguishes the states in P [9]. 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 both a compound distinguishing sequence and a distinguishing set. Moreover, for a given FSM, there may exist a distinguishing set even if no distinguishing sequence exists [2]. Note that distinguishing sets correspond to the so-called adaptive distinguishing sequences in [11], [8]. Finally, notice that a distinguishing set is a compound distinguishing sequence, but the converse does not hold. For instance, a distinguishing set for the FSM in Figure 1 is $\Xi = \{E_1, E_2, E_3, E_4, E_5\}$, with $E_1 = E_2 = aba$, $E_3 = E_4 = E_5 = ab$, which can be derived from the distinguishing sequence aba.

In this paper, we assume that the FSM M is strongly connected, reduced, but not necessarily complete and has a distinguishing set.

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. *An 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 and M are distinguishable, it holds that ω distinguishes them.*

Throughout this paper, $N = (Q, q_0, I, O', D_N, \Delta, \Lambda)$ is an 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 $\{\alpha\}$-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 $\delta(s_0, K) = S$.

III. STATE DISTINGUISHABILITY PROPERTIES

When testing a black box implementation, one faces the problem of determining a state reached when a given test is applied to the implementation. Usually, it is not possible to directly ascertain whether the implementation is in a given state. However, there are conditions which ensure that two sequences take the implementation to a same, but unknown, state. These conditions are captured in the following definition [12].

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, 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 $\delta(s_0, \alpha) = \delta(s_0, \beta)$.*

In other words, considering any FSM with no more states than M that produces the same output response to the given input sequence as M, two sequences in a confirmed set converge in that FSM, if, and only if, they also converge in M. Thus, given an input sequence ω and two sequences of a $\Im_\omega(M)$-confirmed set, we can check whether they lead to the same state in an implementation FSM by simply checking the reached states in the specification FSM. Thus, given a confirmed set K and an input sequence $\alpha \in K$, we refer to it as a *confirmed* sequence. When we need to indicate the final state reached by α then we say that α is *confirmed as state s* $= \delta(s_0, \alpha)$ (in M), or that it is *s-confirmed*. Sequence φ is said to be *verified in state s* (or *s-verified*) if α is *s-confirmed* and $\alpha\varphi$ is *s'-confirmed*, $s' = \delta(s, \varphi)$. We say that the transition (s, x) is *verified* if x is *s-verified*. Given $N \in \Im_\omega(M)$ and state $s \in S$, we denote by $q_s \in Q$ the state reached in N by an *s-confirmed* sequence.

The following theorem provides sufficient conditions for an input sequence to be a checking sequence.

Theorem 1 [14]. *A defined input sequence of a reduced FSM M is a checking sequence for M, if there exists a confirmed set K such that empty sequence is confirmed and each defined transition is verified.*

Using Theorem 1, it is possible to demonstrate that a sequence is a checking sequence by proving the existence of a suitable confirmed set. The problem then is reduced to ensuring that a given set of sequences is confirmed. In [14], we formulated conditions for determining confirmed sequences relying on the identification sequences of a distinguishing set and for extending a given confirmed set with new sequences. These conditions can be summarized in the following rules. Given a sequence ω, we denote by $R(\omega)$ the set of confirmed sequences.

Rule 1: If α is extended (in ω) by the identification sequence E_s, where $s = \delta(s_0, \alpha)$, then add α to $R(\omega)$.

Rule 2: If there exists $\chi \in R(\omega)$ and φ is verified in $\delta(s_0, \chi)$, then add $\chi\varphi$ to $R(\omega)$.

In this paper, we formulate new conditions for extending a confirmed set. To this end, we first introduce the notion of an output-confirmed sequence, i.e., an input sequence for which the output sequence of the implementation is known, and the notion of a convergent set, i.e., a set of sequences which are known to reach the same state in the implementation, even if they are not necessarily confirmed. As we show below, these notions offer new possibilities for ensuring the distinguishability of states reached by sequences and, thus, determining new confirmed sequences.

We say that a sequence φ is *output-confirmed in state s* (or *s-output-confirmed*), if for any $N \in \Im_\omega(M)$, it holds that $\Lambda(q_s, \varphi) = \lambda(s, \varphi)$. For example, any sequence which

extends some *s*-confirmed sequence is also *s*-output-confirmed. Thus, any prefix of the identification sequence E_s is *s*-output-confirmed. Moreover, an *s*-verified sequence is *s*-output-confirmed, but the converse may not hold. Let a sequence α be an *s*-verified sequence and β be an *s'*-output-confirmed, $\delta(s, \alpha) = s'$. Then, it follows that $\alpha\beta$ is *s*-output-confirmed, since $\lambda(s, \alpha\beta) = \lambda(s, \alpha)\lambda(\delta(s, \alpha), \beta) = \lambda(s, \alpha)\lambda(s', \beta) = \Lambda(s, \alpha)\Lambda(s', \beta) = \Lambda(s, \alpha)\Lambda(\Delta(s, \alpha), \beta) = \Lambda(s, \alpha\beta)$.

Two sequences α and β are $\Im_\omega(M)$-*convergent* if for each $N \in \Im_\omega(M)$, it holds that $\Delta(q_0, \alpha) = \Delta(q_0, \beta)$. We say that α and β $\Im_\omega(M)$-*converges*. If it becomes evident that some sequence $\Im_\omega(M)$-converges with another sequence already in a confirmed set, the sequence can be included in the set, yielding thus an extended confirmed set. Moreover, it is obvious that two sequences that are confirmed as the same state are $\Im_\omega(M)$-convergent. The deterministic nature of FSMs considered in this paper implies that if the same sequence extends two $\Im_\omega(M)$-convergent sequences, the resulting sequences are also $\Im_\omega(M)$-convergent. Notice that sequences can be convergent, but not confirmed.

If a sequence α is extended by an *s*-output-confirmed sequence φ which distinguishes s and $\delta(s_0, \alpha)$ then we have that $\Lambda(q_s, \varphi) \neq \Lambda(\Delta(q_0, \alpha), \varphi)$ and, thus, $q_s \neq \Delta(q_0, \alpha)$ for any $N \in \Im_\omega(M)$. Moreover, for any β which $\Im_\omega(M)$-converges to α, it holds that $q_s \neq \Delta(q_0, \beta) = \Delta(q_0, \alpha)$. These observations lead to a lemma that shows how $\Im_\omega(M)$-convergent sequences become confirmed.

Lemma 1. *Let A be a set of $\Im_\omega(M)$-convergent transfer sequences to state s. If, for each state $s' \in S \setminus \{s\}$, there exists an s'-output-confirmed sequence φ which extends some sequence in A and distinguishes s and s', then the sequences in A are s-confirmed.*
Proof. Let $N \in \Im_\omega(M)$. As the sequences in *A* are $\Im_\omega(M)$-convergent, we have that $\Delta(q_0, A) = q$, for some $q \in Q$. It is sufficient to show that $q = q_s$. Let $s' \in S \setminus \{s\}$. Thus, we have that $q_{s'} \in Q \setminus \{q_s\}$. As q and $q_{s'}$ produce different outputs for some sequence φ, it follows that $q \neq q_{s'}$. Hence, $q \notin Q \setminus \{q_s\}$, i.e., $q = q_s$.♦

Traditional means for identifying a state in checking experiments requires that a transfer sequence used to reach the state in question is extended by sequence(s) which distinguish the state from all the other states. The importance of the conditions of Lemma 1 is that proper distinguishing subsequences can in fact extend not necessarily the same but various transfer sequences for the given state. Thus, Lemma 1 offers new possibilities for constructing checking sequences (see Section 5 for examples). Based on this lemma we formulate Rule 3 to be used along with Rules 1 and 2, which indicates when a sequence is confirmed (i.e., added to $R(\omega)$).

Rule 3: Given a transfer sequence α to state *s*, if for each $s' \in S \setminus \{s\}$ there exists an *s'*-output-confirmed sequence φ distinguishing *s* and *s'*, such that φ extends α or a sequence which is $\Im_\omega(M)$-convergent with α, then add α to $R(\omega)$.

The method proposed in Section 4 identifies cases when subsequences with a lesser distinguishability power (and thus possibility shorter than UIO sequences) can be used, and thus exploits new possibilities for shortening checking sequences compared to the existing methods, such as [3], [14] and [7].

IV. GENERATING CHECKING SEQUENCES

In this section we present a method to generate a checking sequence which is based on the notion of confirmed sets and Lemma 1. The basic idea of the method is to consecutively append identification and transfer sequences to a current sequence ω, until each transition of *M* is verified in *K*. Recall that $R(\omega)$ is the set of all confirmed prefixes of ω, which can be obtained by iteratively applying Rules 1-3.

Thus, the method yields a checking sequence by guaranteeing that $R(\omega)$ contains an empty sequence that is confirmed and each defined transition is verified in $R(\omega)$. To check if a transition (s, x) is verified, it suffices to find sequences α and αx in $R(\omega)$, such that $\delta(s_0, \alpha) = s$.

Let ω_i be a sequence obtained in the *i*-th iteration of the method. There are two cases that are dealt with, depending on whether the longest sequence is confirmed or not. The first case occurs when $\omega_i \notin R(\omega_i)$. Then, we identify the longest suffix χ of ω_i and sequence α_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 extended by χ, 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 confirmed. Finally, we have that $\omega_i \in R(\omega_i)$, which is the second case. In this case, we verify a yet unverified transition. 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 the input *x* and a sequence θ, such that $\omega_i\alpha x \in R(\omega_i\alpha x\theta)$ (the sequence θ is chosen based on Lemma 1, as discussed below). The transition (s, x) is, thus, verified w.r.t. $R(\omega_i\alpha x E_{s'})$. The method terminates when no transition remains unverified.

In the simplest case, the sequence θ can be chosen as the identification sequence for the appropriate state. However, Lemma 1 states conditions when proper prefixes of identification sequences are sufficient to determine that a sequence is confirmed. In existing methods for checking sequence generation, these conditions are satisfied by using distinguishing or UIO sequence; see [2], [3], [4], [6], [7], [8], [10], [14], [15], [16], and [17]. Suppose that the transition (s, x) is chosen to be verified in the *i*-th iteration

of the method (thus, $\omega_i \in R(\omega_i)$) and there exists an input sequence α, such that $E_s = x\alpha$, i.e., x is a prefix of E_s. A verified transfer sequence β_i to state s is then chosen, such that $\omega_i\beta_i$ is s-confirmed. The input x is appended, and must be extended by θ which distinguishes $s' = \delta(s, x)$ from each other state. Notice that, in some iteration k, the sequence ω_k is extended by E_s and, consequently, is also s-confirmed. We have that ω_k and $\omega_i\beta_i$ are convergent and so are ω_kx and $\omega_i\beta_ix$. As $\omega_kE_s = \omega_kx\alpha$, the sequence α extends ω_kx. Figure 2 illustrates a possible arrangement of these sequences. Following Lemma 1, it is only necessary that θ distinguishes s' and each state s'', if there is no prefix of α which is s''-output-confirmed and which already distinguishes them.
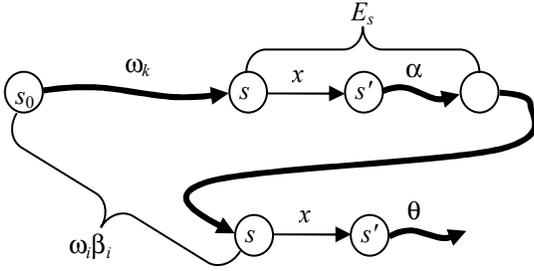

Figure 2: Combined State Distinguishability of Convergent Sequences.

We now present the method for checking sequence generation.

Algorithm 1
Input: A distinguishing set Ξ for an FSM $M = (S, s_0, I, O, D_M, \delta, \lambda)$.
Output: A checking sequence ω
$i \leftarrow 0$
$\omega_0 \leftarrow \varepsilon$
while there exist unverified transitions **do**
   Step 1. if $\omega_i \notin R(\omega_i)$, **then**
- Let $\alpha_i \in \mathit{pref}(\omega_i)$ 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 $\omega_{i+1} \leftarrow \omega_i\varphi$.

   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) is unverified. Let $s' = \delta(s, x)$.
- If x is a prefix of E_s, then let α be such that $E_s = x\alpha$. Determine the shortest prefix θ of $E_{s'}$, such that, for each $s'' \in S \setminus \{s'\}$ which is not distinguishable from s' by any s''-output-confirmed prefix of α, there exists a common prefix of θ and $E_{s''}$ which distinguishes s' and s''. Otherwise, i.e., if x is not a prefix of E_s, let $\theta = E_{s'}$.
- Update $\omega_{i+1} \leftarrow \omega_i\beta_ix\theta$.

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

The algorithm terminates when all transitions are verified. The algorithm starts with Step 1 since $\varepsilon \notin R(\varepsilon)$, to ensure that the empty sequence is confirmed and, by Theorem 1, ω is thus a checking sequence for M. We illustrate the algorithm in Section 6.

Now, we show that the upper bound of a checking sequence length generated by Algorithm 1 is polynomial in the number of states and the number of inputs of M. Let n and t be the numbers of states and transitions of M, respectively. Notice that the proposed algorithm takes advantages of the overlapping between the identification sequences in order to shorten the checking sequence. However, in our analysis, we consider a worst case scenario, where (i) there is no overlapping; (ii) it is not possible to use a prefix of an identification sequence (i.e., Lemma 1 is not applicable), thus, the whole identification sequence is always used and (iii) all identification sequences have the same length l. Suppose that in the i-th iteration of the algorithm, the identification sequence E_s is appended to ω_i, in order to confirm it as state $s = \delta(s_0, \omega_i)$, obtaining $\omega_{i+1} = \omega_iE_s$. If ω_{i+1} is not confirmed yet; in the $(i+1)$-th iteration, Step 1 is executed, appending the identification sequence $E_{s'}$ to confirm ω_{i+1} as state $s' = \delta(s_0, \omega_{i+1})$. Thus, the sequence E_s is now verified in state s. Therefore, after the execution of Step 1 to confirm a sequence as a given state, followed by another execution of Step 1, the identification sequence of that state is verified. Step 1 has to provide verification of at most n identification sequences and can be executed at most $n + 1$ times without the execution of Step 2. On the other hand, Step 2 verifies at least one transition in each execution. Thus, in the worst case, Step 2 can be executed at most t times. Step 2 appends an identification sequence to confirm sequence $\omega_i\beta_ix$ as state $s'' = \delta(s_0, \omega_i\beta_ix)$. If s'' was already verified by Step 1, it follows that $\omega_i\beta_ixE_{s''}$ is already confirmed, i.e., Step 1 will not be executed. Thus, Step 1 is executed at most $n + 1$ times throughout the whole execution of the algorithm. In each execution, it appends the identification sequence of length l. Thus, Step 1 appends at most $(n + 1)l$ inputs to the resulting checking sequence. The execution of Step 2 requires the identification of a (shortest) transfer sequence to the start state of an unverified transition. Such a transfer sequence has at most $n - 1$ inputs. It is extended by an input and the identification sequence. Then, each execution of Step 2 results in appending a sequence of length at most $(n - 1) + 1 + l = n + l$. As Step 2 is executed at most t times, it follows that it produces at most $t(n + l)$ inputs. Thus, a checking sequence generated by Algorithm 1 has at most $(n + 1)l + t(n + l)$ inputs. Thus, the algorithm generates a checking sequence of length $O(nl + nt + tl)$. In [11] an algorithm for finding an adaptive

distinguishing sequence of length $O(n^2)$ is proposed. Thus; replacing l by $O(n^2)$ and t by $O(kn)$, we obtain the upper bound $O(kn^3)$, which coincides with that of the method proposed in [11].

As neither Step 1 nor Step 2 takes exponential time to execute and they are executed at most $n + 1$ and kn times, it follows that the algorithm executes in polynomial time.

V. EXAMPLE

In this section, we present the execution of Algorithm 1 with the FSM in Figure 1 and the distinguishing set $\Xi = \{E_1, E_2, E_3, E_4, E_5\}$, with $E_1 = E_2 = aba$, $E_3 = E_4 = E_5 = ab$. This example is used in previous work [3] and [7] to illustrate global optimization based methods. Thus, the example not only exemplifies the concepts presented in this paper, but also confronts the proposed method with other methods which are based on global optimization. As it will be shown, the use of local optimization leads to better results in this example, what is consistent with the experimental data presented in [14], where a local optimization based method outperforms existing global optimization based methods.

In the first iteration, Step 1 is executed, appending E_1 to $\omega_0 = \varepsilon$. Thus, we have

$$\omega_1 = \omega_0(1)a(2)b(1)a(2)$$

(We indicate the intermediate states in brackets to ease tracing the executions.) Following Rule 1, we determine that the set of confirmed sequences $R(\omega_0)$ contains only the empty sequence. As $\omega_1 \notin R(\omega_1)$, Step 1 is executed again. Then, the shortest prefix α_1 of ω_1 satisfying the conditions of Step 1 is ab, which transfers to state 1. Then, the identification sequence E_1 is appended to ω_1, benefiting from the overlapping between those sequences (the input a). We obtain

$$\omega_2 = \omega_1(2)b(1)a(2)$$

Applying Rule 1, the sequence ab is added to $R(\omega_2)$ (notice that $R(\omega_i) \subseteq R(\omega_{i+1})$). Applying Rule 2, the sequence $abab$ is also added to $R(\omega_2)$, since both ε and ab transfer to state 1 and are in $R(\omega_2)$.

As $\omega_2 \notin R(\omega_2)$, Step 1 is executed again. Now, the shortest prefix of ω_2 which satisfies the conditions of Step 1 is ω_2 itself. As $\delta(s_0, \omega_2) = 2$, the identification sequence E_2 is appended, obtaining

$$\omega_3 = \omega_2(2)a(5)b(3)a(4)$$

Applying Rule 1, the sequence $ababa$ is added to $R(\omega_3)$. Then, the sequence a is also added to $R(\omega_3)$, since both ε and $abab$ are in $R(\omega_3)$ and $\delta(s_0, \varepsilon) = \delta(s_0, abab)$. The sequence aba is added to $R(\omega_3)$ as well. As the sequences ε and a are in $R(\omega_3)$ and $\delta(s_0, \varepsilon) = 1$, the transition $(1, a)$ is verified. Transition $(2, b)$ is also verified, since both a and ab are in $R(\omega_3)$ and $\delta(s_0, a) = 2$.

After executing Step 1 three more times, we have

$$\omega_6 = \omega_3(4)b(5)a(1)b(1)a(2)b(1)a(2)$$

The set of confirmed sequences is $R(\omega_6) = R_3 \cup \{ababaab, ababaabab, ababaababab, ababaababab, ababaabababab, ababaababababa\}$. We have that $\delta(s_0, \omega_6) = 2$ and $\omega_6 \in R(\omega_6)$, i.e., ω_6 is confirmed. Then, Step 2 is executed. The unverified transitions are $(1, b)$, $(2, a)$, $(3, a)$, $(3, b)$, $(4, a)$, $(4, b)$, $(5, a)$ and $(5, b)$. State 2 has an unverified transition; the shortest verified transfer sequence chosen in Step 2 is then the empty sequence and the transition $(2, a)$ is chosen to be verified. We have $\beta_6 = \varepsilon$. As $\delta(2, a) = 5$, we have to extend $\omega_6\beta_6a$ with a sequence θ which distinguishes state 5 from each other state. As input a is a prefix of E_2 and is extended by input b, the states which are distinguished from 5 by input b and in which input b is output-confirmed do not need to be distinguished by θ. In this case, input b is only output-confirmed in state 2, but it does not distinguish states 2 and 5. Thus, no proper prefix of E_5 satisfies the conditions required for θ and, thus, the whole E_5 is used. We obtain

$$\omega_7 = \omega_6(2)a(5)a(1)b(1)$$

Applying Rule 1, we have that ω_6a is added to $R(\omega_7)$; the transition $(2, a)$ is indeed verified. Thus, by Rule 2, the sequence $ababaa$ is also added. As the sequence $ababaab$ is already in $R(\omega_7)$ and $\delta(s_0, ababaa) = 5$, the transition $(5, b)$ is verified as a consequence of the verification of transition $(2, a)$, The sequence ω_7 transfers the FSM M to state 1 and is confirmed. Thus, Step 2 is executed again. The unverified transitions are $(1, b)$, $(3, a)$, $(3, b)$, $(4, a)$, $(4, b)$ and $(5, a)$. As state 1 has an unverified transition, the shortest verified transfer sequence is the empty sequence. Thus, the transition $(1, b)$, $\delta(1, b) = 1$, is chosen to be verified. As b is not a prefix of E_1, θ becomes the identification sequence E_1. Then, we have

$$\omega_8 = \omega_7(1)b(1)a(2)b(1)a(2)$$

After applying Rules 1 and 2, the sequences ω_7b, ω_7ba, ω_7bab and ω_7baba are added to $R(\omega_8)$ and the transition $(1, b)$ becomes verified. The unverified transitions are $(3, a)$, $(3, b)$, $(4, a)$, $(4, b)$ and $(5, a)$. The shortest verified sequence which transfers to a state with unverified transition is $\beta_8 = a$, which transfers to state 5. Then, the transition $(5, a)$, $\delta(5, a) = 1$, is chosen to be verified. Input a is a prefix of the identification sequence E_5. Input b is output-confirmed in states 2 and 5, but it does not distinguish state 1 from them. Thus, sequence θ is the identification sequence E_1. After appending the transfer sequence, the input a and θ, we obtain

$$\omega_9 = \omega_8(2)a(5)a(1)a(2)b(1)a(2)$$

Applying Rules 1 and 2, the sequences ω_8a, ω_8aa, ω_8aaa, ω_8aaab and ω_8aaaba are added to $R(\omega_9)$. The list of unverified transitions is now reduced to $(3, a)$, $(3, b)$, $(4, a)$ and $(4, b)$, since the transition $(5, a)$ is also verified. The sequence ω_9 is confirmed and $\delta(s_0, \omega_9) = 2$. The shortest

verified transfer sequence to a state with unverified transitions is $\beta_9 = ab$, which transfers to state 3. Either transition $(3, a)$ or $(3, b)$ might be chosen. The transition $(3, a)$ is chosen to be verified, $\delta(3, a) = 4$. (In this case, the transition was chosen due to lexicographical order: transition $(3, a)$ precedes transition $(3, b)$. As a matter of fact, if the transition $(3, b)$ would have been chosen, the length of the resulting checking sequence would be increased by 3). Input a is a prefix of E_3. Input b is output-confirmed in states 1, 2, and 5 and distinguishes state 4 from each of them. Thus, sequence θ only needs to distinguish states 4 and 3, which is done by the prefix $\theta = a$ of E_4. Then, instead of using the whole E_4, $\theta = a$ is appended to verify the transition $(3, a)$. We have

$$\omega_{10} = \omega_9(2)a(5)b(3)a(4)a(4)$$

After applying Rules 1 and 2, the sequences ω_9a, ω_9ab and ω_9aba are added to $R(\omega_{10})$. Notice that transitions $(3, a)$ and $(4, b)$ are now verified. The unverified transitions are $(3, b)$ and $(4, a)$. As ω_9 is not confirmed, Step 1 is executed, appending E_4, since $\delta(s_0, \omega_{10}) = 4$, and obtaining

$$\omega_{11} = \omega_{10}(4)a(4)b(5)$$

The application of Rules 1 and 2 results in the addition of the sequences $\omega_{10}a$ and $\omega_{10}ab$ to $R(\omega_{11})$. Thus, the transition $(4, a)$ is verified. The only remaining unverified transition is now $(3, b)$. The shortest verified transfer sequence to state 3 is $\beta_{11} = b$. As input b is not a prefix of E_3, sequence θ must be the identification sequence E_5. After appending β_{11}, input b and E_5, we obtain

$$\omega_{12} = \omega_{11}(5)b(3)b(5)a(1)b(1)$$

The resulting checking sequence of length 36 is shorter than checking sequences constructed by the existing methods. The method given in [7] was reported to generate a checking sequence of length 64. The method proposed in [3] is reported to generate a checking sequence of length 44. The method proposed in [14] produces a sequence of length 43.

VI. Related Work

There has been much interest in the generation of checking sequences. Gonenc [4] presents a method, known as the method D, for generation of checking sequences, which is divided into two parts. In the first part, an α-sequence is generated, such that the distinguishing sequence is guaranteed to distinguish the states in the implementation. In the second part, a β-sequence is generated, such that each transition is verified. A β-sequence is a concatenation of transitions, followed by the distinguishing sequence, using transfer sequences, if necessary. The α- and β-sequences are then concatenated to form a checking sequence.

Kohavi and Kohavi [10] 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. The notion of distinguishing sets corresponds to adaptive distinguishing sequences; their use in checking sequence generation has been also discussed recently [8].

Ural *et al.* [16] propose a method that attempts to minimize the length of checking sequence generated using distinguishing sequence and graph-theoretical modeling. Sufficient conditions for a sequence to be a checking sequence are formulated there and used in several work, e.g., [6], [3], [15], [7], and [17]. The α- and β-sequences of Gonenc's method are divided in [16] 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 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 in [16] 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 in [6] and [7].

Based on [16], Chen *et al.* [3] demonstrate that the verification of the last transition traversed by a distinguishing sequence applied to a particular state is implied by the verification of the other transitions. This possibility is also exploited in [4]. The authors present an algorithm that identifies transitions whose explicit verification can be skipped.

The possibility of overlapping the distinguishing sequence, as proposed by Boute [2], is exploited by Ural and Zhang in [15]. 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.

Yalcin *et al.* [17] state sufficient conditions under which UIO sequences may be used instead of a distinguishing sequence. Although that work represents a step towards in relaxing conditions for using sequences with lesser state distinguishability power than distinguishing sequences and sets to verify a state, the conditions presented in [17] can rarely be satisfied, limiting the use of UIO sequences. In this paper, we presented weaker conditions, which allow using UIO sequences for checking sequence generation when the existing conditions do not, resulting in a bigger reduction on its length. Now we demonstrate that the corresponding theorem in that work is a special case of Lemma 1. Given an input sequence α, let $symb(\alpha)$ be the set of input symbols that appear in α. Let U_s be an arbitrary UIO sequence for state s, not necessarily related to distinguishing sets (we reserve E_s to represent a UIO sequence from a

distinguishing set). The main theorem in [17], using the notions proposed in this works, would state that if for each state $s' \in S$ and input $x \in symb(U_s)$, it holds that x is s'-verified and a sequence α, $\delta(s_0, \alpha) = s$, is extended by U_s, then both α and αU_s are confirmed. Notice that if for each state $s' \in S$ and input $x \in symb(U_s)$, x is s'-verified, then U_s is s'-verified (and, thus, s'-output-confirmed). Thus, it can be proved that this theorem is a special case of Lemma 1. Hence the method proposed in [17] could also benefit from Lemma 1 by employing weaker conditions for using UIO sequences. The work [17] reports an example FSM for which the method presented in this paper reduces the length of the checking sequence from 26 to 13.

The possibility of using UIOs in checking sequence generation is also investigated in [13]. The proposed method uses a compound distinguishing sequence to form a so-called γ-sequence. This sequence ensures that the sequences which are UIOs for the specification FSM are also UIOs for the implementation. The use of UIOs in this way, however, does not seem to pay off, as indicated by the experiments described in [13]. The method generated checking sequences even longer than those generated by Gonenc's method [4], when the specification has a distinguishing sequence. The work reports an example with a checking sequence of length 80. The method proposed in this paper generates a checking sequence of length 36 for the same example.

In our previous work [14], we elaborated an approach which uses the notion of confirmed sequences as input sequences convergent in all FSMs which react to a given test sequence as the specification FSM. The approach allowed us to relax the previously known sufficient conditions [16] for a sequence to be a checking sequence. The resulting method uses a local optimization approach which diverges from methods proposed in recent work (namely, [16], [6], [3], [7], [15], and [17]), where graph-theoretical modeling is used for minimizing the length of checking sequence. The main disadvantage of the graph-theoretical approaches is that they 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.* [16], while a local optimization based approach attempts determining them during the construction of checking sequence. Experimental results [14] indicate that the latter may deliver shorter sequences and thus deserve more investigation. In this paper, we further develop a local optimization based approach of [14]; in particular, we formulated conditions allowing using sequences with less state distinguishability power than distinguishing and UIO sequences to verify states in constructing checking sequences. We use these conditions to elaborate a checking sequence generation method.

VII. CONCLUSION

This paper investigated possibilities of using subsequences of a lesser state distinguishability power than distinguishing and UIO sequences for shortening checking sequences.

The main contributions of this paper are as follows. Firstly, the existing conditions which allow the use of UIO sequences in checking sequence generation are relaxed, providing a ready improvement to the existing method which employs both, distinguishing and UIO, sequences. Secondly, it is demonstrated that state verification can be achieved not only with distinguishing and UIO sequences, but also with sequences with even less state distinguishability power. Finally, we also presented a polynomial time method that exploits new possibilities for shortening checking sequences. Using an example, we showed that the proposed method can generate shorter checking sequences than existing methods.

As future work, we can envisage the following possible extensions of the presented results. Firstly, we showed that the proposed method can lead to shorter checking sequences. However, experimental studies are necessary to understand the extent of this shortening. As the proposed method is based on a local optimization, local improvements may not always result in global gains, and more studies are necessary to find a compromise between the scope of optimization and complexity in constructing checking sequences. Finally, there are checking sequences which do not satisfy the proposed conditions. Thus, there is still room for shortening 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 vol. 39(11), pp. 1604–1615, 1991.

[2] Boute, R.T.: Distinguishing sets for optimal state identification in checking experiments. IEEE Transactions on Computers, vol. 23(8), pp. 874–877, 1974.

[3] Chen, J., Hierons, R.M., Ural, H., Yenigun, H.: Eliminating redundant tests in a checking sequence. In: TestCom 2005. Number 3502 in LNCS, pp. 146–158, 2005.

[4] Gonenc, G.: A method for the design of fault detection experiments. IEEE Transactions on Computers, vol 19(6), pp. 551–558, 1970.

[5] Hennie, F.C.: Fault-detecting experiments for sequential circuits. In: Proceedings of Fifth Annual Symposium on Circuit Theory and Logical Design, 95–110, pp. 1965.

[6] Hierons, R.M., Ural, H.: Reduced length checking sequences. IEEE Transactions on Computers, vol. 51(9), pp. 1111–1117, 2002.

[7] Hierons, R.M., Ural, H.: Optimizing the length of checking sequences. IEEE Transactions on Computers, vol. 55(5), pp. 618–629, 2006.

[8] Hierons, R., Jourdan, G-V., Ural, H., Yenigun, H. Using adaptive distinguishing sequences in checking sequence constructions. In: ACM SAC 2008, pp. 682–687, 2008.

[9] Hsieh, E. P.: Checking Experiments for Sequential Machines. IEEE Transactions on Computers vol. 20(10), pp. 1152–1166, 1971.

[10] Kohavi, I., Kohavi, Z.: Variable-length distinguishing sequences and their application to the design of fault-detection experiments. IEEE Transactions on Computers, vol 17(8), 792–795, 1968.

[11] Lee, D., Yannakakis, M.: Testing finite-state machines: state identification and verification. IEEE Transactions on Computers, vol 43(3), 306–320, 1994.

[12] Simão, A., Petrenko, A. Checking FSM test completeness based on sufficient conditions. CRIM-07/10-20, Montreal, Quebec, Canada, 2007.

[13] B. Serdar, T. Kuo-Chung: A New A New Approach To Checking Sequence Generation for Finite State Machines, In TestCom 2002, IFIP Conference Proceedings; Vol. 210, pp. 391-404, 2002.

[14] Simão, A., Petrenko, A.: Generating checking sequences for partial reduced finite state machines. In TestCom 2008. Number 5047 in LNCS, pp. 153–168, 2008.

[15] Ural, H., Zhang, F.: Reducing the lengths of checking sequences by overlapping. In: TestCom 2006. Number 3964 in LNCS, pp. 274–288, 2008.

[16] Ural, H., Wu, X., Zhang, F.: On minimizing the lengths of checking sequences. IEEE Transactions on Computers, vol. 46(1), pp. 93–99, 1997.

[17] Yalcin, M.C., Yenigun, H.: Using distinguishing and uio sequences together in a checking sequence. In: TestCom 2006. Number 3964 in LNCS pp. 259–273, 2006.