# Checking Completeness of Tests for Finite State Machines

Adenilso Simao and Alexandre Petrenko

**Abstract**—In testing from a Finite State Machine (FSM), the generation of test suites which guarantee full fault detection, known as complete test suites, has been a long-standing research topic. In this paper, we present conditions that are sufficient for a test suite to be complete. We demonstrate that the existing conditions are special cases of the proposed ones. An algorithm that checks whether a given test suite is complete is given. The experimental results show that the algorithm can be used for relatively large FSMs and test suites.

**Index Terms**—Finite State Machine, test analysis, fault coverage, test completeness conditions, test generation.

## 1 Introduction

Test generation from a Finite State Machine (FSM) is a long-standing research problem, with numerous contributions over decades. Since the seminal work of Moore [12] and Hennie [8], several methods have been proposed to generate a test suite with full fault detection capability, i.e., a test suite which provides full coverage of the set of all possible FSMs with a certain number of states that model implementations of a given specification FSM; such test suites have complete fault coverage and, in this sense, are complete [1], [2], [4], [5], [9], [10], [15], [17], [18], [20]. These methods rely on sufficient conditions for test suite completeness. The conditions appear either explicitly in the methods or implicitly in the proof of their correctness.

The generation methods usually require the existence of sequences which identify states in the specification FSM based on their outputs. If the FSM is completely specified and has a diagnostic sequence, a complete test suite with a single sequence can be generated, as in, e.g., [5], [9], [10], [8], [18]. The sufficient conditions underlying the correctness proof of these methods are captured in a theorem presented in [18]. However, a diagnostic sequence may not exist for an arbitrary reduced FSM. In this case, methods which do not require the existence of a diagnostic sequence can be used, such as those presented in [17], [20]. These methods are applicable to any reduced FSMs and generate test suites with multiple sequences, as they rely on the availability of a reliable reset operation. The related sufficient conditions are summarized in [14] and refined in [2].

Besides supporting the definition of generation methods, sufficient conditions for test completeness can be used to address other related issues, namely, the analysis of the fault coverage of a test suite and test minimization. Completeness of a test suite can be established by exhaustive approaches which explicitly enumerate either all possible faulty FSMs, as in, e.g., [16] or all minimal forms of the partially specified FSM representing a test suite as a tree (see [19], [6]). By their nature, these approaches do not scale well. This fact explains why approaches which reduce the task of deciding whether a given test suite has complete fault detection capability to checking the satisfaction of sufficient conditions appear to be more practical even if they cannot give a definitive answer when the conditions are not satisfied.

The relevance of investigating sufficient completeness conditions is thus twofold. On one hand, weakening sufficient conditions can allow for improvement in methods for test generation, obtaining shorter tests of a proven fault detection capability. On the other hand, weaker sufficient conditions can be used to prove completeness of a much larger class of tests, as well as to further minimize existing complete tests.

In this paper, we present sufficient conditions for test suite completeness that are weaker than the ones known in the literature. We consider the case when implementation FSMs have at most as many states ($n$) as the specification FSM. Test completeness in this case is usually called $n$-completeness. We introduce the notion of *confirmed* sequence set. A set of input sequences is confirmed with respect to a test suite $T$ and an FSM $M$ if sequences leading to a same state in $M$ also lead to a same state in any FSM that has the same output responses to $T$ and has as many states as $M$. We show that if there exists a confirmed set which includes the empty sequence and traverses each defined transition, then a test suite is $n$-complete. We also demonstrate that the proposed conditions generalize both those proposed in [18] (which do not need a reliable reset but require a diagnostic sequence) and in [2] (which need a reliable reset but do not require a diagnostic sequence). We also present an approach for determining confirmed sets and elaborate an algorithm for analyzing test completeness. The effectiveness of the algorithm is demonstrated by experimenting with randomly generated FSMs with up to 500 states and test suites with up to 300,000 inputs.

• *A. Simao is with the Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Av. Trabalhador São-carlense, 400—Centro, PO Box 668, São Carlos 13560-970, SP, Brazil.*
*E-mail: adenilso@icmc.usp.br.*
• *A. Petrenko is with the Centre de Recherche Informatique de Montreal (CRIM), 405, Avenue Ogilvy, Bureau 101, Montreal, Quebec H3N 1M3, Canada. E-mail: petrenko@crim.ca.*

*Manuscript received 24 Sept. 2007; revised 26 May 2008; accepted 26 Oct. 2009; published online 14 Jan. 2010.*
*Recommended for acceptance by S. Tragoudas.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2007-09-0480.*
*Digital Object Identifier no. 10.1109/TC.2010.17.*

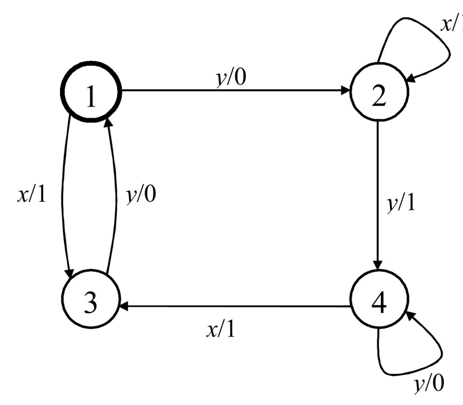

Fig. 1. A partial FSM with the initial state 1.

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, state sufficient conditions for a test suite to be $n$-complete, based on the existence of confirmed sets and elaborate an approach for determining confirmed sets. An algorithm for checking $n$-completeness is presented in Section 4. We then demonstrate in Section 5 that all known sufficient conditions for $n$-completeness are special cases of the conditions proposed in this paper. The results of the experimental evaluation of the formulated conditions and method to check them are discussed in Section 6. Section 7 concludes the paper.

## 2 Definitions

A Finite State Machine is a deterministic Mealy machine, which can be defined as follows:

**Definition 1.** *A* Finite State Machine *(FSM) M is a 7-tuple $(S, s_0, I, O, D, \delta, \lambda)$, 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,*
- *$D \subseteq S \times I$ is a specification domain,*
- *$\delta : D \rightarrow S$ is a transition function, and*
- *$\lambda : D \rightarrow O$ is an output function.*

If $D = S \times I$, then $M$ is a *complete* FSM; otherwise, it is a *partial* FSM. As $M$ is deterministic, a tuple $(s, x) \in D$ determines uniquely a (*defined*) *transition* of $M$ in state $s$. For simplicity, we use $(s, x)$ to denote the transition, thus omitting its output and final state. A string $\alpha = x_1 \dots x_k,\ \alpha \in I^*$, is said to be a *defined* input sequence for state $s \in S$, if there exist $s_1, \dots, s_{k+1}$, where $s_1 = s$, such that $(s_i, x_i) \in D$ 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 $\Omega_M$ 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. Fig. 1 shows the example of a partial FSM.

We extend the transition and output functions from input symbols to defined input sequences, including the empty sequence $\varepsilon$, as usual: for $s \in S, \delta(s, \varepsilon) = s$ and $\lambda(s, \varepsilon) = \varepsilon$; and for input sequence $\alpha$ and input $x$, $\delta(s, \alpha x) = \delta(\delta(s, \alpha), x)$ 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 state $s$ of $M$, and a set of defined input sequences $C \subseteq \Omega(s)$, 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 $\delta(s, C) = \{s'\}$. Given sequences $\alpha, \beta, \gamma \in I^*$, if $\beta = \alpha\gamma$, then $\alpha$ is a *prefix* of $\beta$; if, moreover, $\gamma$ is not empty, then $\alpha$ is a *proper* prefix of $\beta$. A set of sequences $A$ is *prefix-closed*, if for each sequence $\beta \in A$, it holds that $A$ contains all prefixes of $\beta$.

An FSM $M$ is said to be *initially connected*, if for each state $s \in S$, there exists a defined input sequence $\alpha \in \Omega_M$, called a *transfer* sequence for state $s$, such that $\delta(s_0, \alpha) = s$. In this paper, only initially connected machines are considered, since any state that is not reachable from the initial state can be removed without changing the machine's behavior.

Two states $s, s' \in S$ are *distinguishable*, if there exists $\gamma \in \Omega(s) \cap \Omega(s')$, such that $\lambda(s, \gamma) \neq \lambda(s', \gamma)$. We say that $\gamma$ *distinguishes* $s$ and $s'$. If a sequence $\gamma$ distinguishes each pair of distinct states, then $\gamma$ is a *diagnostic* sequence. 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 finally 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.

## 3 Complete Test Suite and Sufficient Conditions

We consider only deterministic machines in this paper. Thus, a test case can be defined using just inputs, as expected outputs are uniquely determined from the inputs by a given specification FSM.

**Definition 2.** *A defined input sequence of FSM M is called a* test case *(or simply a* test*) of M. A* test suite *T of M is a finite prefix-closed set of tests of M. A test $\alpha \in T$ is* maximal *(with respect to T), if it is not a proper prefix of another test in T.*

The execution of a test implies the execution of all its proper prefixes. Thus, to execute a test suite only its maximal tests have to be considered. As tests should be applied in the initial states, the implementation must be brought to its initial state before the application of a test. If the test suite possesses only a single maximal test, this can be accomplished by using a homing sequence, as in [8], [9], [18]. On the other hand, to execute a test suite with more than one maximal test, it is assumed that the implementation has a reset which reliably brings the machine to its initial state prior to applying the next test, e.g., [1], [2], [4], [14]. For the sake of simplicity, we define the length of a test $\alpha$ as $|\alpha| + 1$, i.e., the number of inputs plus a reset needed to bring the machine to the initial state, regardless of the fact that test suites with a single maximal test does not actually require such a reset. The length of a test suite $T$ is defined as the sum of the lengths of all its maximal tests.

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

**Definition 3.** *A given test suite T of FSM M is* $n$-complete, *if for each FSM $N \in \Im(M)$, such that N and M are distinguishable, there exists a test in T that distinguishes them.*

If an $n$-complete test suite is the set of all prefixes of a single sequence $R \in \Omega_M$, i.e., $R$ is its only maximal test, then

$R$ is, in fact, a so-called "checking sequence," used for testing FSM without a reset operation [18].

In this paper, we are concerned with conditions that are sufficient to guarantee that a given test suite is $n$-complete. We first introduce the notion of confirmed sets of defined input sequences. Throughout this paper, let $N = (Q, q_0, I, O', D', \Delta, \Lambda)$, where $D' = Q \times I$, be an arbitrary element of $\Im(M)$. Given a test suite $T$, let $\Im_T(M)$ be the set of all $N \in \Im(M)$, such that $N$ and $M$ are $T$-equivalent.

**Definition 4.** *Let T be a test suite of an FSM $M = (S, s_0, I, O, D, \delta, \lambda)$ and $K \subseteq T$. The set K is* confirmed *if $\delta(s_0, K) = S$ and, for each $N \in \Im_T(M)$, 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 words, a confirmed set of input sequences contains transfer sequences for all states of $M$ and any sequences converging (i.e., leading to a same state) in any FSM that has the same output responses to $T$ and has as many states as $M$ also converge in $M$. This key property is exploited by methods for constructing complete test suites, such as [1], [2], [4], [5], [9], [10], [8], [15], [17], [18], [20], in one way or another.

Notice that, according to Definition 4, we can establish that two sequences in a confirmed set for a given test suite $T$ converge in any FSM that reacts to $T$ as the FSM $M$ only by determining that they converge in the FSM $M$.

The next theorem states that, for a given test suite to be $n$-complete for given FSM, it suffices that there exists a confirmed set which contains the empty sequence and covers each transition of FSM. A set of input sequences *covers* a transition if the set contains a transfer sequence for its initial state and the sequence is extended in the confirmed set with the input labelling the transition.

**Theorem 1 (Sufficient Conditions for $n$-Completeness of a Test Suite).** *Let T be a test suite of an initially connected reduced FSM $M = (S, s_0, I, O, D, \delta, \lambda)$ with n states. T is* $n$-complete *for M, if there exists a confirmed set $K \subseteq T$ with the following properties:*

1. $\varepsilon \in K$.
2. *For each $(s, x) \in D$, there exist $\alpha, \alpha x \in K$, such that $\delta(s_0, \alpha) = s$.*

**Proof.** Let $N \in \Im_T(M)$. As $M$ is initially 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 we have that $\Delta(q_0, \beta) \neq \Delta(q_0, \alpha)$. Thus, $|Q| = n$. Consequently, there exists a bijection $f : S \rightarrow Q$, 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 $\nu$, and, moreover, $\lambda(s, x) = \Lambda(f(s), x)$, for each $(s, x) \in D$.

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 $f(\delta(s_0, \varphi)) = \Delta(q_0, \varphi)$. There exists $\alpha \in K$, such that $\delta(s_0, \alpha) = \delta(s_0, \varphi)$ and $\alpha x \in K$. Thus, we have that $f(\delta(s_0, \alpha x)) = \Delta(q_0, \alpha x)$ and $\Delta(q_0, \alpha) = f(\delta(s_0, \alpha)) = f(\delta(s_0, \varphi)) = \Delta(q_0, \varphi)$. It follows that

$$\begin{aligned} f(\delta(s_0, \varphi x)) &= f(\delta(\delta(s_0, \varphi), x)) = f(\delta(\delta(s_0, \alpha), x)) \\ &= f(\delta(s_0, \alpha x)) = \Delta(q_0, \alpha x) = \Delta(\Delta(q_0, \alpha), x) \\ &= \Delta(\Delta(q_0, \varphi), x) = \Delta(q_0, \varphi x). \end{aligned}$$

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 $(s, x) \in D$, there exists $\alpha x \in T$, $\delta(s_0, \alpha) = s$, $\alpha \in K$. 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 $T$-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 defined 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 $\alpha x \in K$, such that $\lambda(\delta(s_0, \alpha), x) = \Lambda(f(\delta(s_0, \alpha)), x)$. From $\delta(s_0, \alpha) = \delta(s_0, \nu)$, it follows that $\lambda(\delta(s_0, \nu), x) = \Lambda(f(\delta(s_0, \nu)), x) = \Lambda(\Delta(q_0, \nu), x)$; and from $\lambda(s_0, \nu) = \Lambda(q_0, \nu)$, it follows that

$$\begin{aligned} \lambda(s_0, \nu x) = \lambda(s_0, \nu)\lambda(\delta(s_0, \nu), x) &= \Lambda(q_0, \nu)\Lambda(\Delta(q_0, \nu), x) \\ &= \Lambda(q_0, \nu x). \end{aligned}$$

The resulting contradiction concludes the proof. □

If all the sequences in $T$ are prefixes of a single input sequence, the test can be applied without a reliable reset. Thus, the conditions apply to both testing scenarios, with and without a reliable reset operation. In Section 5, we show that they are weaker than those known in the literature in either case.

The following lemmas indicate several possibilities for constructing a confirmed set. Our first lemma presents a sufficient condition for a minimal state cover (which contains a single transfer sequence for each state) to be a confirmed set. Given a test suite $T$ of an FSM $M$, two sequences $\alpha, \beta \in T$ are *T-distinguishable* (or simply distinguishable), if there exist $\alpha\gamma, \beta\gamma \in T$, such that $\lambda(\delta(s_0, \alpha), \gamma) \neq \lambda(\delta(s_0, \beta), \gamma)$.

**Lemma 1.** *Let T be a test suite of FSM M and K be a minimal state cover. If each two sequences of K are T-distinguishable, then K is confirmed.*

**Proof.** Let $N \in \Im_T(M)$. The set $K$ contains exactly $n$ transfer sequences for all states of $M$, then, for each $s \in S$, there exists only one sequence in $K$ that takes $M$ to $s$. For any $\alpha, \beta \in K$, we have that $\delta(s_0, \alpha) \neq \delta(s_0, \beta)$ and $\Delta(q_0, \alpha) \neq \Delta(q_0, \beta)$. Therefore, as $N$ has no more states than $M$, we have that $|\Delta(q_0, K)| = n$, and $K$ is confirmed. □

The next statements indicate sufficient conditions for adding a sequence to a set while preserving the property of "being confirmed" of the set, based on which confirmed sets can incrementally be derived.

**Lemma 2.** *Let K be a confirmed set and $\alpha$ be a transfer sequence for state s. If for each $s' \in S \setminus \{s\}$, there exists $\beta \in K$, $\delta(s_0, \beta) = s'$, such that $\alpha$ and $\beta$ are T-distinguishable, then the set $K \cup \{\alpha\}$ is confirmed.*

**Proof.** Let $N \in \Im_T(M)$. Let $f : S \rightarrow Q$ be a bijection, such that for each $\chi \in K, f(\delta(s_0, \chi)) = \Delta(q_0, \chi)$. It is sufficient to show that $f(s) = \Delta(q_0, \alpha)$. For each $s' \in S \setminus \{s\}$, there exists $\beta \in K, \delta(s_0, \beta) = s'$, such that $\alpha$ and $\beta$ are $T$-distinguishable. Therefore, we have that $\Delta(q_0, \alpha) \neq \Delta(q_0, \beta) = f(s')$. It follows that $\Delta(q_0, \alpha) = f(s)$ and, thus, $K \cup \{\alpha\}$ is confirmed. □

The next statement relies on the fact that, if proper prefixes of some transfer sequences converge, then the sequences converge as well.

**Lemma 3.** *Let $K$ be a confirmed set and $\alpha \in T$. If there exist $\beta, \chi \in K$, such that $\delta(s_0, \beta) = \delta(s_0, \chi)$, and a sequence $\varphi$, such that $\beta\varphi \in K$ and $\chi\varphi = \alpha$, then the set $K \cup \{\alpha\}$ is also confirmed.*

**Proof.** Let $N \in \Im_T(M)$. As $\chi, \beta$ and $\beta\varphi$ are in $K$, we have that $\Delta(q_0, \chi) = \Delta(q_0, \beta)$ and, therefore, it follows that

$$\Delta(q_0, \beta\varphi) = \Delta(\Delta(q_0, \beta), \varphi) = \Delta(\Delta(q_0, \chi), \varphi) = \Delta(q_0, \chi\varphi) = \Delta(q_0, \alpha).$$

Thus, as $\delta(s_0, \beta\varphi) = \delta(s_0, \alpha)$ and $\Delta(q_0, \beta\varphi) = \Delta(q_0, \alpha)$, we have that $K \cup \{\alpha\}$ is confirmed. □

In the following theorem, we summarize the above lemmas in sufficient conditions for a given set of defined input sequences to be confirmed:

**Theorem 2 (Sufficient Conditions for the Existence of a Confirmed Set).** *Let $T$ be a test suite of FSM $M$ with $n$ states and $L \subseteq T$ be a set of $k$ sequences, $k \geq n$. For an arbitrary ordering of the sequences $\alpha_1, \ldots, \alpha_k$ in $L$, let $L_i = \{\alpha_j \in L \mid 1 \leq j \leq i\}$. Then $L$ is a confirmed set if there exists an ordering $\alpha_1, \ldots, \alpha_k$, such that the corresponding $L_1, \ldots, L_k$ satisfy the following conditions:*

1. *$L_n$ is a minimal state cover such that every two sequences are T-distinguishable.*
2. *If $k > n$, then for each $\alpha_i, n < i \leq k$, it holds that either*
    a. *for each $s \in S \setminus \{\delta(s_0, \alpha_i)\}$, there exists $\beta \in L_{i-1}, \delta(s_0, \beta) = s$, such that $\alpha_i$ and $\beta$ are T-distinguishable; or*
    b. *there exist $\chi$, $\beta$, and $\varphi$, such that $\alpha_i = \chi\varphi$, and $\beta\varphi$, $\beta$, $\chi \in L_{i-1}$, $\delta(s_0, \beta) = \delta(s_0, \chi)$;*

**Proof.** We prove by induction on $L_i$. For the basis step, $L_n$ is a confirmed set by Lemma 1. For the induction step, assume that $L_i$, $n \leq i < k$, is a confirmed set. We show that $L_{i+1}$ is also confirmed. If 2a holds, then Lemma 2 applies; otherwise, if 2b holds, Lemma 3 does. Consequently, the set $L_i \cup \{\alpha_i\} = L_{i+1}$ is confirmed. □

## 4 Algorithm for Checking $n$-Completeness

In this section, we present an algorithm for determining the $n$-completeness of a given test suite based on Theorems 1 and 2. As the conditions of these theorems are sufficient, if the algorithm terminates with a positive result, then the test suite is indeed $n$-complete. However, as the conditions are not necessary, based on a negative answer, we cannot conclude that the test suite is not $n$-complete.

The algorithm involves three main steps:

1. minimal confirmed sets are identified by applying Lemma 1 to a given test suite $T$;
2. the minimal confirmed sets are repeatedly extended by the application of Lemmas 2 and 3 to sequences of $T$ as long as possible, thus obtaining maximal confirmed sets; and
3. the maximal confirmed set are checked for satisfaction of Theorem 1.

We first apply Lemma 1 to find minimal confirmed sets (i.e., containing a single transfer sequence for each state of $M$), which are subsets of $T$ with $n$ pairwise $T$-distinguishable sequences. The problem of finding minimal confirmed sets can be cast as a problem of finding cliques in a graph, as follows: We define a *distinguishability graph* $G$ on $T$ as a graph whose vertices are the sequences in $T$, such that two vertices are adjacent in $G$ if and only if the corresponding sequences are $T$-distinguishable. Then, the sequences that appear in a clique of size $n$ (an $n$-clique) of $G$ form a confirmed set. The problem of finding $n$-cliques in an arbitrary graph is NP-complete [11]. However, several properties of distinguishability graphs can be used to formulate heuristics which allow dealing with large graphs. Notice first that $G$ is an $n$-partite graph, since the sequences that transfer to same state are not adjacent and, therefore, we can partition its vertices into $n$ blocks. Thus, we deal with the special case of finding $n$-cliques in an $n$-partite graph. This problem has already been investigated in [7], where a specialized algorithm is proposed to find all $n$-cliques. The algorithm implements a branch-and-bound approach, where a partial solution is extended in a search tree (branching), and the search is pruned as soon as it is possible to determine that a given partial solution is fruitless (bounding). The initial partial solution is a trivial empty clique. It is extended with sequences that are adjacent to every sequence in the partial clique. Based on the fact that the graph is $n$-partite, the authors propose heuristics that help determine very early when a partial clique cannot be extended to an $n$-clique. The proposed heuristics are also useful to solve our problem. Moreover, differently from that work, we do not need to find all $n$-cliques, as discussed below.

From a minimal confirmed set $K$, we can obtain a confirmed set $K' \subseteq T$, such that $K \subseteq K'$ and $K'$ is the largest set which satisfies the conditions in Theorem 2. To determine $K'$, we initialize a set $K_{cur}$ (a current confirmed set) with $K$. Then, we iteratively select a sequence $\alpha \in T \setminus K_{cur}$ and try to apply either Lemma 2 or Lemma 3. If no new sequence satisfies them, the confirmed set $K_{cur}$ so far obtained is the largest one.

Notice that it is not necessary to check a minimal confirmed set $K$ that is included in some largest confirmed set $K'$ that was already analyzed, as stated in the next lemma.

**Lemma 4.** *Let $K$ be a largest confirmed set that satisfies the conditions of Theorem 2. Let $K'$ be a minimal confirmed set and $K''$ be the largest confirmed set obtained by applying Lemmas 2 and 3 to the set $K'$. Then if $K' \subseteq K$, it holds that $K'' \subseteq K$.*

**Proof.** We prove by contradiction. Assume that $K' \subseteq K$, and $K'' \nsubseteq K$. The sequences of $K''$ can be ordered as $\alpha_1, \ldots, \alpha_k$, according to Theorem 2. Let $j$ be such that $K_j = \{\alpha_1, \ldots, \alpha_{j-1}\} \subseteq K$, but $\alpha_j \notin K$. Thus, there exists a set of sequences $W \subseteq K_j$ which, in conjunction with $\alpha_j$, satisfy the conditions of either Lemma 2 or Lemma 3. In this case, $K$ can be extended by the inclusion of $\alpha_j$, since $W \subseteq K$. However, this contradicts the fact that $K$ is a largest set with respect to the conditions of Theorem 2. □

Thus, according to Lemma 4, after finding an $n$-clique that represents a minimal confirmed set, the search tree can be bounded whenever it can be concluded that any $n$-clique obtained from a given partial clique would be included in some largest confirmed set already determined. As discussed in Section 6, this heuristic is a key factor for the performance of the algorithm.

It remains to verify whether the obtained maximal confirmed set satisfies Theorem 1, which is a straightforward step. If it does, the test suite is $n$-complete for $M$. Otherwise, if another minimal confirmed set can be found, the whole process iterates again.

We finally present the algorithm in details.

**Algorithm 1.**

**Input**: An FSM $M$ and a test suite $T$.

**Output**: **True**, if $T$ is $n$-complete according to Theorems 1 and 2.

```
1. Build the distinguishability graph $G$ of $T$.
2. Let $L$ be the empty set.
3. Determine (by using the branch-and-bound approach)
   an $n$-clique $K$ of $G$, such that there does not exist
   $K' \in L$ with $K \subseteq K'$. If no such a clique exists, then
   terminate with the answer False.
4. Find a sequence $\alpha \in T \setminus K$, such that either Lemma 2
   or Lemma 3 can be applied. If no such a sequence
   exists, go to Step 6.
5. Include $\alpha$ in $K$ and go to Step 4.
6. If $K$ satisfies Theorem 1, then terminate with the
   answer True.
7. Include $K$ in $L$ and go to Step 3.
```

We have implemented Algorithm 1 in a tool and applied the tool to randomly generated FSMs and test suites. The obtained results are discussed in Section 6.

## 5 Comparison with Previous Work

In this section, we show that the sufficient conditions in [2] and [18] are special cases of the conditions presented in this paper. The conditions of [18] are formulated for checking sequences; moreover, they can only be applied to strongly connected FSMs possessing a diagnostic sequence, which may not exist for an arbitrary reduced FSM. The conditions of [2] apply to initially connected FSMs even without diagnostic sequences; however, an implementation is assumed to have a reliable reset operation and, thus, test suites can have more than one maximal test.

In [2], the authors present the weakest sufficient conditions found in literature for an $n$-complete test suite using the reset operation. The conditions are stated in Theorem 3, slightly rephrased using our notations.

**Theorem 3 [2].** *Let T be a test suite and Q be a prefix-closed state cover of an FSM M, such that the following conditions hold:*

1. *For all sequences $\alpha, \beta \in Q$, such that $\delta(s_0, \alpha) \neq \delta(s_0, \beta)$, it holds that $\alpha$ and $\beta$ are T-distinguishable.*
2. *For each defined transition $(s, x) \in D$, there exists $\alpha x \in T$, such that $\delta(s_0, \alpha) = s$, with the following properties.*
   a. *For each $\beta \in Q$, such that $\delta(s_0, \beta) \neq s$, it holds that $\alpha$ and $\beta$ are T-distinguishable.*
   b. *For each $\beta \in Q$, such that $\delta(s_0, \beta) \neq \delta(s, x)$, it holds that $\alpha x$ and $\beta$ are T-distinguishable.*

*Then, T is n-complete.*

We show that Theorem 3 is a special case of Theorem 1.

**Theorem 4.** *Let T be a test suite as in Theorem 3. Then T satisfies the conditions of Theorem 1.*

**Proof.** We first show that a state cover $Q \subseteq T$ defined in Theorem 3 is a confirmed set. Let $Q_0 \subseteq Q$ be a minimal state cover. Clearly, each two sequences in $Q_0$ are $T$-distinguishable, by Condition 1. Then, by Lemma 1, $Q_0$ is a confirmed set. Let $\nu \in Q \setminus Q_0$. By Condition 1, $\nu$ is $T$-distinguishable from each sequence $\alpha \in Q_0$ which does not transfer to the same state as $\nu$. Consequently, by Lemma 2, we have that $Q_0 \cup \{\nu\}$ is a confirmed set and so is $Q$. Condition 2.i implies that $Q \cup \{\alpha\}$ is a confirmed set, since $\alpha$ is $T$-distinguishable from each sequence $\beta \in Q$ which does not transfer to the same state as $\alpha$ and, therefore, Lemma 2 can be applied. Similarly, $Q \cup \{\alpha x\}$ is a confirmed set, as $\alpha x$ is $T$-distinguishable from each sequence $\beta \in Q$ which does not lead to the same state as $\alpha x$. Thus, let $K$ be a confirmed set which includes $Q$ and the corresponding sequences $\alpha$ and $\alpha x$, for each defined transition $(s, x) \in D_M, \delta(s_0, \alpha) = s$. Therefore, as $\varepsilon \in Q$ (recall that $Q$ is prefix-closed), $K$ satisfies the conditions of Theorem 1. □

We now demonstrate that the converse is not true, by showing an $n$-complete test suite for which Theorem 3 does not hold, while Theorem 1 does. Consider the FSM in Fig. 1 and the test suite $T_1$ which contains $xyyxy$, $yyyyyyxyyy$, and their prefixes. It does not satisfy the conditions of Theorem 3, since there is no state cover in the test suite $T_1$ with the required characteristics. Indeed, *xyy* is the only sequence which leads to state 2 and is followed by $x$ in $T_1$. Therefore, $\alpha = xyy$ is the only sequence that could be used in Condition 2 for the defined transition (2, $x$). However, the input sequence *xy* is the only sequence applied after the sequence *xyy*, but it does not distinguish state 2 from state 3, since input $x$ is not defined in latter state. Thus, Condition 2i is violated.

Nonetheless, by using Lemmas 1, 2, and 3, we can find a confirmed set, which satisfies the conditions of Theorem 1. We have that the set $\{\varepsilon, y, yy, yyyyyyx\} = K_0$ is confirmed, by Lemma 1. By repeatedly applying Lemma 2, we can prove that the set $K_0 \cup \{xyyx, yyy, yyyyyyxy, yyyyyyxyy\} = K_1$ is confirmed. After several applications of Lemma 3, we obtain the confirmed set $K_1 \cup \{yyyy, yyyyy, yyyyyy\} = K_2$. Now, we can apply Lemma 2 to prove that $K_2 \cup \{x\} = K_3$ is confirmed. Finally, we add sequences *xy* and *xyy* by applying Lemma 3 and obtain the confirmed set $\{\varepsilon, x, xy, xyy, xyyx, y, yy, yyy, yyyy, yyyyy, yyyyyy, yyyyyyx, yyyyyyxy\}$, which satisfies the conditions of Theorem 1.

Ural et al. [18] present conditions for a sequence to be a checking sequence. In that work, a sequence is defined to be a checking sequence if it can distinguish a complete strongly connected deterministic reduced FSM $M$ from each FSM with at most as many as states as $M$ that is not isomorphic to $M$. The conditions rely on the existence of a diagnostic sequence (also called a distinguishing sequence). We first restate a definition used in [9], [10], [18] for constructing checking sequences for complete as well as partial reduced FSMs.

**Definition 5.** *Given $R \in \Omega_M$, let $d$ be a diagnostic sequence of a strongly connected deterministic reduced (possibly partial) FSM M. Then,*

1. *A prefix $\alpha$ of R is (d-)recognized in R if $\alpha d$ a prefix of R.*
2. *If $\alpha, \beta$, and $\alpha\gamma$ are recognized in R and $\delta(s_0, \alpha) = \delta(s_0, \beta)$, then $\beta\gamma$ is recognized in R.*
3. *If $\alpha$ and $\alpha x$ are recognized in R and $\delta(s_0, \alpha) = s$, then the transition (s, x) is verified in R.*

We present a theorem which is similar to the one formulated in [18], but is stronger in the sense that all the implementation FMSs which are distinguishable from the specification FSM are considered, and not only those which are not isomorphic [18] or not equivalent to the specification FSM [9], [10]. The statement is a special case of Theorem 1 and, thus, takes into account initialization faults as well, as opposed to [9], [10], [18]. Compared to the original version of the theorem, we add the requirement that $d$ must be a prefix of the checking sequence.

**Theorem 5.** *Given $R \in \Omega_M$, if $d$ is a prefix of R and every transition of M is verified in R, then the set of prefixes of R is an n-complete test suite.*

**Proof.** Let $K_0$ be the set of $d$-recognized prefixes of $R$. We first show that $\delta(s_0, K_0) = S$. Let $s \in S$. There exists at least one recognized sequence that leads to $s$, since every transition is verified and $M$ is strongly connected. For a sequence to be recognized, either Condition 1 or Condition 2 must hold. For Condition 2, however, another recognized sequence that also leads to $s$ is required and, consequently, at least one sequence satisfies Condition 1. Therefore, for each $s$, there exists at least one sequence that is $d$-recognized and, thus, $s \in \delta(s_0, K_0)$.

As $d$ is a diagnostic sequence, for all $\alpha, \beta \in K_0$, such that $\delta(s_0, \alpha) \neq \delta(s_0, \beta)$, it holds that $\alpha$ and $\beta$ are $T$-distinguishable. Then, by Lemma 1, $K_0$ is a confirmed set. Furthermore, we have that $\varepsilon \in K_0$, since $d$ is a prefix of $R$. If $\alpha, \beta, \alpha\gamma$, and $\beta\gamma$ are prefixes of $R$ and $\alpha, \beta$, and $\alpha\gamma$ are in a confirmed set, then $\beta\gamma$ can also be included in the confirmed set, by Lemma 2. Consequently, if a sequence $\varphi$ is recognized and $K'$ is a confirmed set, then so is $K' \cup \{\varphi\}$. Let $K$ be the set of all recognized prefixes of $R$. It follows that $K$ is a confirmed set and $K_0 \subseteq K$. As every transition is verified in $R$, for each $(s, x) \in D$, there exist $\alpha, \alpha x \in K$. Therefore, by Theorem 1, the set of $R$'s prefixes is $n$-complete. □

We now present an example of an $n$-complete test suite that satisfies Theorem 1, but not Theorem 5. Consider the FSM in Fig. 1 and the sequence $R = yyyyyyxyyyxyxyyxy$. The shortest diagnostic sequence for this FSM is $yyy$. The $d$-recognized sequences are $\varepsilon$, $y$, $yy$, $yyy$, and $yyyyyyx$. The recognized sequences are $yyyy$, $yyyyy$, and $yyyyyy$. Then, the set of verified transitions is $\{(1, y), (2, y), (4, y), (4, x)\}$, which includes only four out of seven defined transitions.

Now we demonstrate that the test suite $T_2$, containing the maximal test $R = yyyyyyxyyyxyxyyxy$, satisfies Theorem 1 and, thus, that $R$ is a checking sequence. First, it holds, by Lemma 1, that $\{\varepsilon, y, yy, yyyyyyx\} = K_0$ is a confirmed set. By the application of Lemma 2, we have that $K_0 \cup \{yyy\} = K_1$ is confirmed. We repeatedly apply Lemma 3 to prove that $K_1 \cup \{yyyy, yyyyy, yyyyyy\} = K_2$ is a confirmed set. Using Lemma 2, we obtain the confirmed set $K_2 \cup \{yyyyyyxy, yyyyyyxyy\} = K_3$. Then, $K_3 \cup \{yyyyyyxyyy, yyyyyyxyyyx, yyyyyyxyyyxy\} = K_4$ is a confirmed set, according to Lemma 3. Next, we have that $K_4 \cup \{yyyyyyxyyyxyx\} = K_5$ is also confirmed (Lemma 2). Now, we can prove that $K_5 \cup \{yyyyyyxyyyxyxy, yyyyyyxyyyxyxyy\} = K_6$ is a confirmed set. Finally, the sequences $yyyyyyxyyyxyxyyx$ and $yyyyyyxyyyxyxyyxy$ are also confirmed according to Lemmas 2 and 3, respectively. The resulting confirmed set satisfies the conditions of Theorem 1.

Another interesting feature of the conditions is that they are more flexible than the previous ones. For instance, although both the test suites $T_1$ and $T_2$ satisfy conditions proposed in this paper, they do not satisfy the conditions of [2] and [18]. The test suite $T_1$ has length 17 and two maximal tests, whereas the test suite $T_2$ has length 18 and a single maximal test. Thus, the proposed conditions are not parameterized with the number of resets needed to execute all the tests; this feature allows to elaborate a test generation method to produce a test suite which is most suitable (in terms of the number of tests) to a given situation.

Another approach to determine whether a given test suite is $n$-complete is presented in [19], [14]. Given an FSM $M$ and a test suite $T$, the tree machine with the set of defined sequences being exactly $T$ is first constructed. Then one needs to construct all the possible reduced forms of the tree machine (the FSM $M$ is one of them), using an existing algorithm for partial FSM minimization (recent publications on this topic include, e.g., [6], [13]). If at least one of the obtained reduced FSMs is distinguishable from $M$, then $T$ is not $n$-complete. Otherwise, it is $n$-complete. Compared to our approach, this method is exhaustive, while ours is approximate, in the sense that we can positively identify some $n$-complete test suites, but cannot provide definitive negative answer. However, the problem of partial FSM minimization is NP-complete and the existing algorithms can deal only with small machines and small test suites, as the experimental results of recent publications (e.g., [6]) show. Our method must also deal with the NP-complete problem of finding an $n$-clique. Nonetheless, the heuristics derived from Lemma 4 and the fact that the distinguishability graph is $n$-partite allow us to cope with significantly larger FSMs and test suites (compared to [6], [19]), as the experimental results in Section 6 indicate. On the other hand, the solution of [6] requires that all $n$-cliques be found and checked, so its applicability is reduced to FSMs with few states.

## 6 Experimental Results

To evaluate the proposed sufficient conditions as well as the method for checking test completeness a number of experiments involving random generation of FSMs and tests were performed using a tool, called `Chico` (**Ch**ecking **co**mpleteness), which checks whether test suites satisfy the conditions proposed in this paper. The first set of experiments addressed the scalability of the method, and the second compared it with the checkers of conditions of [2] and [18].

TABLE 1
Number of Minimal Confirmed Sets and Size of the First Found Largest Confirmed Sets

| Number of States | Number of Minimal Confirmed Sets | Size of the First Found Largest Confirmed Set |
|---|---|---|
| 4 | 3900 | 157 |
| 5 | 3888 | 172 |
| 6 | 12321 | 158 |
| 7 | 216178 | 184 |
| 8 | 1206465 | 192 |
| 9 | 620544 | 184 |
| 10 | 654750 | 182 |

In the experiments, we used randomly generated FSMs and test suites. We randomly generate initially connected reduced FSMs 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, if needed, more transitions (by randomly selecting two states, an input, and an output) to the machine until the required (given a priori) number of transitions is obtained. In the third phase, the distinguishability of each pair of distinct states is checked. If the FSM is not reduced, it is discarded and another FSM is generated.

Once a reduced FSM is obtained, a test suite is randomly generated as follows: We start with a test suite $T_{cur}$ containing only the empty sequence, i.e., $T_{cur} = \{\varepsilon\}$. Then, a defined sequence $\alpha$ is iteratively generated starting from $\alpha = \varepsilon$ by adding to it an input randomly selected among those defined in the state reached by the current sequence. The sequence growing process terminates as soon as $\alpha \notin T_{cur}$; the sequence $\alpha$ is then included into $T_{cur}$. After the inclusion of $\alpha$, the number of sequences in $T_{cur}$ is increased by one.

## 6.1 Scalability of the Proposed Algorithm

An important question is how many minimal confirmed sets have to be analyzed for a given test suite. To answer this question, we executed Chico with the FSM in Fig. 1 and 10,000 randomly generated test suites. We observed that the tool usually finds the first minimal confirmed set rather quickly and the maximal confirmed set is then determined. The subsequent search for another minimal confirmed set is bounded quickly due to Lemma 4. In this experiment, no test suite required the analysis of more than two minimal confirmed sets, and in most cases, only a single minimal confirmed set was analyzed. Moreover, only in 144 out of 10,000 test suites, two minimal confirmed sets were used. This experiment indicates that the number of minimal confirmed sets to be analyzed may not be always

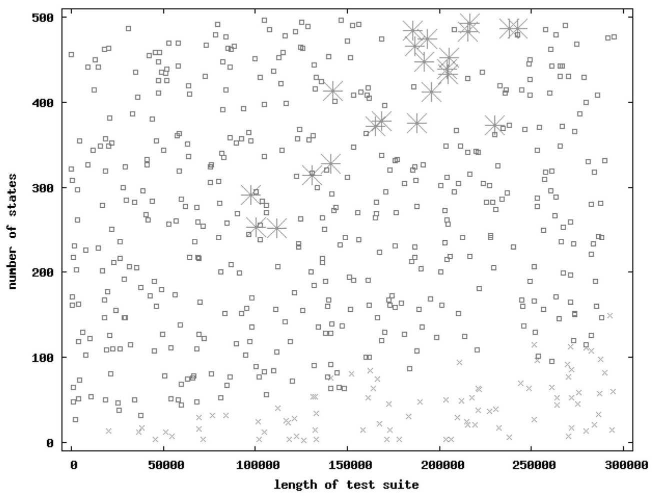

Fig. 2. Distribution of runs.

large in spite of the fact that their total number grows exponentially with the number of states. This dependency is an essential impediment to any approach explicitly enumerating all $n$-cliques of a graph, e.g., [19]. However, for our algorithm, the larger the number of $n$-cliques, the easier it is to find one of them and the remaining search can be bounded early. Table 1 illustrates the saving due to Lemma 4 in another set of experiments. We randomly generated reduced complete FSMs with two inputs, two outputs, and test suites of with 200 tests and selected the FSMs for which the number of minimal confirmed sets is the largest, representing a worst-case scenario. For none of them, the test suite was determined to be $n$-complete by the tool. Indeed, the number of minimal confirmed sets is large (see, for instance, the experiments with the FSM with eight states). However, the size of the largest confirmed set obtained from the first identified minimal confirmed set is also large. Then, all other minimal confirmed sets are included in the first largest confirmed set and this fact can be established rather early, bounding the search.

During some of the experiments with large FSMs and tests, the runtime to find the first minimal confirmed set becomes unacceptably long. This is not surprising, since the problem is NP-complete and even with the heuristics employed in the tool it may eventually take an exponential amount of time to find a minimal confirmed set. An important question here is how often the tool fails due to the impossibility of finding a minimal confirmed set in a reasonable amount of time. We have chosen a timeout of one hour to terminate executions. All the experiments were run on a Pentium IV HT 64 bits 3.4 GHz computer, with 2 Gb of memory. We generated 500 FSMs with 10 inputs, 10 outputs, number of states randomly chosen between one and 500 as well as 500 test suites of length between one and 300,000. Fig. 2 shows the results, where small crosses represent runs that ended before the timeout expiration with a positive answer (the test suite was $n$-complete), small squares represent runs that ended before the timeout with a negative answer, and big stars represent the ones lasting at least one hour. There were 22 runs terminated by the timeout, which correspond to 4.4 percent of the executions;

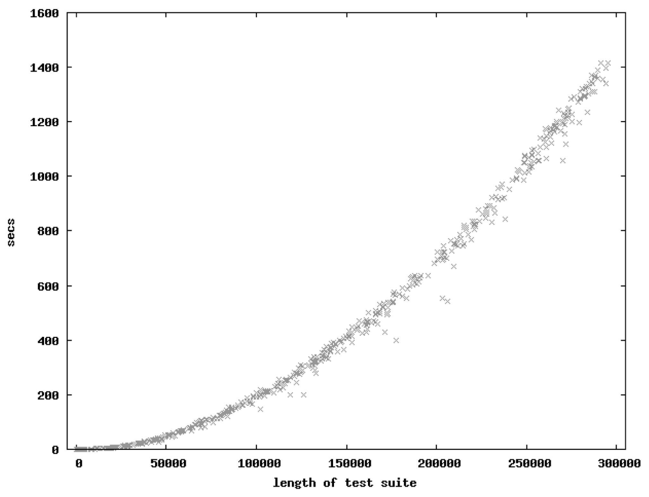

Fig. 3. Execution time variation with the length of a test suite.

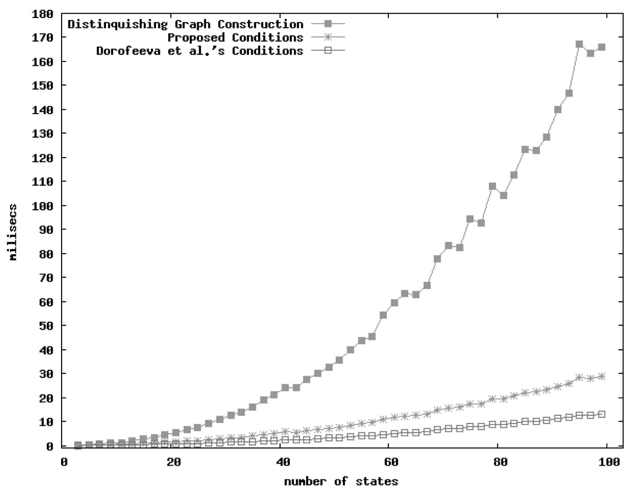

Fig. 4. Average execution time for checking the proposed conditions and the conditions from [2] for complete test suites.

none of them occurred for FSMs with fewer than 200 states or for test suites with length smaller than 80,000.

To see how execution time depends on the number of states of the FSMs, we randomly generated 500 reduced complete FSMs with 10 inputs, 10 outputs, and states ranging from 3 to 500, as well as test suites with 20,000 tests. We consider only the runs that were not ended by a timeout. As seen in the previous experiment, the probability of a run ending by timeout for this setting is negligible, since timeouts only occurred with larger test suites. The average time was 61.046 seconds and the standard deviation was 3.451 seconds. Thus, all things being equal, the execution time varies only slightly with the number of states. Actually, we observed that the parameter with the greatest impact on execution time is the test suite length, as discussed next.

Fig. 3 shows how the execution time grows as the test suite length increases. We generated 500 complete FSMs with 10 inputs, 10 outputs, and the number of states ranging from 3 to 500. The length of the test suites ranges from 1 to 300,000. Since the number of edges in the distinguishability graph and, consequently, the time for constructing it, grows quadratically with the test suite length, the overall execution time increases in the same way. We notice that even for test suites of length as big as 300,000 and for FSMs with up to 500 states, the tool was able to produce a result in less than 1,500 seconds. In this experiment, we also excluded the runs in which the tool was terminated by timeout. For larger test suites, the tool runs out of memory, since the amount of memory required for data structures used to build and represent the distinguishability graph also grows quadratically with the length of the test suite.

## 6.2 Experimental Comparison with Previous Work

The conditions proposed in this paper are more complex than the conditions in previous work, except for [19]. Therefore, an important question is what is the overhead of their checking. We compare the scalability of methods checking the proposed and existing sufficient conditions. Notice that neither [2] nor [18] discuss how the conditions can be checked, since $n$-complete test suite generation is the focus of either work. Nonetheless, checkers for both conditions could easily be derived from the tool `Chico`, by limiting the application of certain lemmas and heuristics.

For Dorofeeva et al.'s conditions, Lemmas 3 and 4 are not applicable and, moreover, the use of Lemma 2 is limited to prefix-closed confirmed sets. We implemented a checker of the conditions of [2], named `ChicoD`, and determined the time required to check the $n$-completeness of test suites in two scenarios.

In the first scenario, complete reduced FSMs are randomly generated, but the test suites are obtained by the State Counting method [15], which produces $n$-complete test suites. This scenario is the most favorable for the conditions of [2], since a test suite obtained by the above method satisfies them and, moreover, the determination of a suitable state cover required by those conditions is straightforward. Not surprisingly, all things being equal, `Chico` needs more time than `ChicoD`. In Fig. 4, we present the average execution time for 100 FSMs with three inputs, three outputs, and number of states ranging from 3 to 100, totalling 9,800 FSMs. For each FSM, we generated an $n$-complete test suite using State Counting method. Notice that the length of the test suite increases as the number of states increases. We divided the execution time into two parts: the time required to construct the distinguishing graph of a given test suite and the time for checking the respective conditions based on the graph. The results show that, although `Chico` employs more complex conditions, the overhead is still reasonable, even when the conditions of [2] can be more promptly checked. Notice that as the FSMs grow (and, consequently, the length of the test suite increases), the time required to construct the distinguishing graph increases faster than the time required to check the conditions. The distinguishing graph allows one to avoid recalculating the $T$-distinguishability of pairs of sequences and, thus, cannot be removed without an increase in the execution time. We, thus, observe that the overhead tends to become insignificant, even in the scenario that is most favorable to Dorofeeva et al.'s conditions.

In the second scenario, we used randomly generated test suites. In this scenario, the heuristics that we implemented in `Chico` allow treating much larger FSMs than the checker

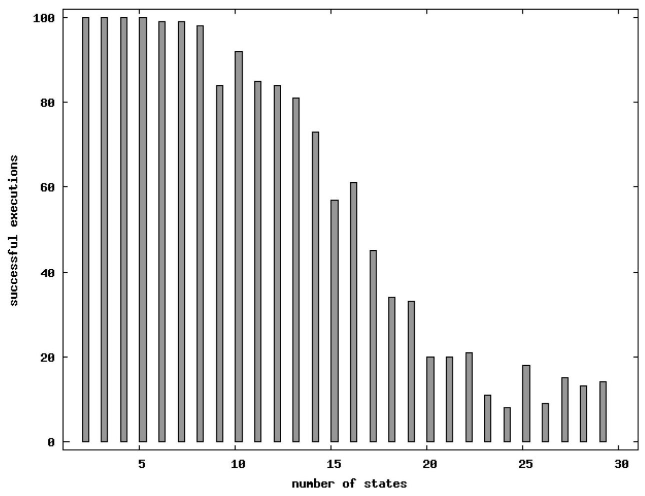

Fig. 5. Percentage of executions of ChicoD which terminate before timeout for random test suites.

of Dorofeeva et al.'s conditions. We randomly generated 100 complete reduced FSMs with three inputs, three outputs, and number of states from 3 to 30. The test suites are randomly generated with 1,000 test cases. Each execution of `Chico` terminated in less than 0.2 seconds. On the other hand, the execution of `ChicoD` took often more than 10 minutes, which is a timeout we set to terminate executions. Fig. 5 shows the number of executions that terminate before the timeout. Notice that while for small FSMs the conditions of Theorem 3 could be checked for every FSM, the probability that `ChicoD` fails to verify the Dorofeeva et al.'s conditions within a reasonable amount of time grows with the state number of a specification FSM, since Lemma 4 is not applicable and state covers to verify are numerous.

To check whether a given input sequence satisfies Theorem 5, it is first necessary to determine if any of its prefixes is a diagnostic sequence. Then, only this diagnostic sequence is considered to determine $T$-distinguishability of sequences. However, the actual limitation of those conditions is their applicability. For instance, experimental studies [3] indicate that the probability of a randomly generated FSM to have a diagnostic sequence is low, circa 15 percent. Moreover, even if an FSM has a diagnostic sequence, the probability that an input sequence satisfies the conditions of Theorem 5 is yet smaller. We implemented a checker of the conditions of [18], named `ChicoU`. Lemmas 1, 2, and 3 are still applicable in checking those conditions, but Lemma 4 is not. We generated 100 complete reduced FSMs with three inputs, three outputs, and number of states from 3 to 15. Then, we randomly generated input sequences of length 1,000. The test suite obtained from each input sequence is then checked with `Chico`. If Theorem 1 is not satisfied, the FSM and test suite are discarded and others are generated. We repeat this process until we obtain a set of 1,200 FSMs (i.e., 100 for each size of FSMs) and respective checking sequences that satisfy Theorem 1. Then, using `ChicoU`, we checked whether they also satisfy Theorem 5. We observed that the number of checking sequences satisfying Theorem 5 drops quickly. For FSMs with three states, 88 out of 100 sequences satisfy it, while for FSMs with eight states only two sequences do. For bigger FSMs, no checking sequence satisfies Theorem 5.

The experimental results obtained for relatively large FSMs and tests indicate that the proposed conditions have a wider applicability compared to [18]; checking them scales better than checking the conditions [19] and [2].

## 7 Conclusions

In this paper, we presented sufficient conditions for test suite $n$-completeness that are weaker than those known in the literature. The conditions apply to both testing scenarios, with and without reliable reset operation. They can be used in several ways. On one hand, sufficient conditions can guide the definition of new generation methods or the improvement of existing ones. Elaboration of such a method based on the proposed sufficient conditions is an open research issue. On the other hand, the $n$-completeness of existing test suites can be checked by the algorithm we proposed. Strategies for minimizing complete tests without losing fault detection capability can also be elaborated. Although the algorithm requires the identification of a clique in a graph, an NP-complete problem, the experimental results we presented show that the algorithm can be used for relatively large FSMs and test suites.

As future work, we can mention several possible extensions of the presented results. First, it is interesting to see how Theorem 1 can be extended to the case of $m$-completeness, where $m \geq n$. Another possible generalization of conditions would be to consider nondeterministic specification FSMs. Finally, since the proposed test completeness conditions are only claimed to be sufficient, we believe that the quest for necessary and sufficient conditions will go on.

## Acknowledgments

This work was in part supported by the Natural Sciences and Engineering Research Council of Canada under discovery grant OGP0194381 and by Brazilian Funding Agency CNPq under grant 200032/2008-9. The authors wish to thank the reviewers for their useful comments.

## References

[1] T.S. Chow, "Testing Software Design Modeled by Finite-State Machines," *IEEE Trans. Software Eng.*, vol. 4, no. 3, pp. 178-187, May 1978.

[2] R. Dorofeeva, K. El-Fakih, and N. Yevtushenko, "An Improved Conformance Testing Method," *Formal Techniques for Networked and Distributed Systems*, pp. 204-218, Springer, 2005.

[3] R. Dorofeeva, N. Yevtushenko, K. El-Fakih, and A.R. Cavalli, "Experimental Evaluation of FSM-Based Testing Methods," *Proc. Third IEEE Int'l Conf. Software Eng. and Formal Methods*, pp. 23-32, 2005.

[4] S. Fujiwara, G.v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test Selection Based on Finite State Models," *IEEE Trans. Software Eng.*, vol. 17, no. 6, pp. 591-603, June 1991.

[5] G. Gonenc, "A Method for the Design of Fault Detection Experiments," *IEEE Trans. Computers*, vol. 19, no. 6, pp. 551-558, June 1970.

[6] S. Gören and F.J. Ferguson, "On State Reduction of Incompletely Specified Finite State Machines," *Computers & Electrical Eng.*, vol. 33, no. 1, pp. 58-69, 2007.

[7] T. Grunert, S. Irnich, H.-J. Zimmermann, M. Schneider, and B. Wulfhorst, "Finding All k-Cliques in k-Partite Graphs: An Application in Textile Engineering," *Computers & Operations Research*, vol. 29, pp. 13-31, 2002.

[8] F.C. Hennie, "Fault-Detecting Experiments for Sequential Circuits," *Proc. Fifth Ann. Symp. Circuit Theory and Logical Design,* pp. 95-110, 1964.

[9] R.M. Hierons and H. Ural, "Reduced Length Checking Sequences," *IEEE Trans. Computers,* vol. 51, no. 9, pp. 1111-1117, Sept. 2002.

[10] R.M. Hierons and H. Ural, "Optimizing the Length of Checking Sequences," *IEEE Trans. Computers,* vol. 55, no. 6, pp. 618-629, May 2006.

[11] R.M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Computations,* R.E. Miller and J.W. Thatcher, eds., pp. 85-103, Springer, 1972.

[12] E.P. Moore, "Gedanken-Experiments," *Automata Studies,* C. Shannon and J. McCarthy, eds., Princeton Univ. Press, 1956.

[13] J.M. Pena and A.L. Oliveira, "A New Algorithm for Exact Reduction of Incompletely Specified Finite State Machines," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 11, pp. 1619-1632, Nov. 1999.

[14] A. Petrenko, G.v. Bochmann, and M. Yao, "On Fault Coverage of Tests for Finite State Specifications," *Computer Networks and ISDN Systems,* special issue on protocol testing, vol. 29, pp. 81-106, 1996.

[15] A. Petrenko and N. Yevtushenko, "Testing from Partial Deterministic FSM Specifications," *IEEE Trans. Computers,* vol. 54, no. 9, pp. 1154-1165, Sept. 2005.

[16] J.F. Poage and E.J. McCluskey Jr., "Derivation of Optimal Test Sequences for Sequential Machines," *Proc. IEEE Fifth Symp. Switching Circuits Theory and Logical Design,* pp. 121-132, 1964.

[17] M.P. Vasilevskii, "Failure Diagnosis of Automata," *Cybernetics,* vol. 4, pp. 653-665, 1973.

[18] H. Ural, X. Wu, and F. Zhang, "On Minimizing the Lengths of Checking Sequences," *IEEE Trans. Computers,* vol. 46, no. 1, pp. 93-99, Jan. 1997.

[19] M. Yao, A. Petrenko, and G.v. Bochmann, "Fault Coverage Analysis in Respect to an FSM Specification," *Proc. IEEE INFOCOM '94,* pp. 768-775, 1994.

[20] N. Yevtushenko and A. Petrenko, "Synthesis of Test Experiments in Some Classes of Automata," *Automatic Control and Computer Sciences,* vol. 24, no. 4, pp. 50-55, 1990.

**Adenilso Simao** received the BS degree in computer science from the State University of Maringa (UEM), Brazil, in 1998, and the MS and PhD degrees in computer science from the University of Sao Paulo (USP), Brazil, in 2000 and 2004, respectively. Since 2004, he has been a professor of computer science at the Computer System Department of USP. From August 2008 to July 2010, he has been on a sabbatical leave at Centre de Recherche Informatique de Montreal (CRIM), Canada. His research interests include software testing and formal methods. He is a member of the Brazilian Computer Society (SBC).

**Alexandre Petrenko** received the diploma degree in electrical and computer engineering from Riga Polytechnic Institute in 1970 and the PhD degree in computer science from the Institute of Electronics and Computer Science, Riga, USSR, in 1974. He was also awarded other degrees and titles, namely, "Doctor of Technical Sciences" and "Senior Research Fellow in Technical Cybernetics and Information Theory" from the Supreme Attestation Committee, Moscow, USSR, and "Doctor Habil. of Computer Science" from the Latvian Scientific Council, Riga, Latvia. Until 1992, he was head of a computer network research lab of the Institute of Electronics and Computer Science in Riga. From 1979 to 1982, he was with the Computer Network Task Force of the International Institute for Applied Systems Analysis (IIASA), Vienna, Austria. From 1992 to 1996, he was a visiting professor/researcher of the Universite de Montreal. He joined the Centre de Recherche Informatique de Montreal (CRIM) in 1996, where he is currently a senior researcher and team leader. In 2005, along with C. Campbell, M. Veanes, and J. Huo, he received the best paper award from the 17th IFIP International Conference on Testing of Communicating Systems. He has published more than 150 research papers and has given numerous invited lectures worldwide. He is a member of the IFIP TC6 Working Group 6.1 "Architectures and Protocols for Distributed Systems" and serves as a member of the program committee for a number of international conferences and workshops. He is a member of the steering committee of the IFIP International Conference on Testing of Communicating Systems (TestCom). His current research interests include formal methods and their application in distributed systems and computer networks.