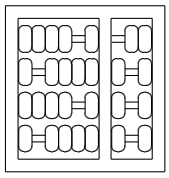

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Partial FSM Models and Completeness with Blocking Test Cases

Adilson Luiz Bonifacio *Arnaldo Vieira Moura*

Technical Report - IC-13-33 - Relatório Técnico

November - 2013 - Novembro

The contents of this report are the sole responsibility of the authors.
O conteúdo do presente relatório é de única responsabilidade dos autores.

Partial FSM Models and Completeness with Blocking Test Cases

Adilson Luiz Bonifacio* Arnaldo Vieira Moura†

Abstract

Test suite generation and coverage analysis have been widely studied for FSM-based models. Several studies focused on specific conditions for verifying completeness of test suites. Some have found necessary conditions for test suite completeness, whereas other approaches obtained sufficient, but not necessary, conditions for this problem. Most of these works restricted the specification or the implementation FSM models in several ways. More recent works have described necessary and sufficient conditions for test suite completeness, but only under the assumption that all implementations be complete FSM models. In this work we describe necessary and sufficient conditions for test suite completeness under more relaxed constraints. We also show an intrinsic relation between the number of states in implementations and complete test suites. Further we show necessary and sufficient conditions for test suite completeness in the presence of blocking sequences in both the specification and the implementation models.

1 Introduction

Many studies have investigated the automatic generation of test suites based on Finite State Machine (FSM) models. Several of them focused on the automatic generation of test suites with full fault detection. In other words, they provide test suites with complete fault coverage [2, 3, 4, 5, 9, 10, 11, 14, 17]. Several of these methods have shown sufficient conditions that guarantee the completeness of the test suites. Some other works proved necessary conditions [12, 18] for the completeness of test suites.

However, in most of them, specifications are required to be reduced with n states, while the corresponding implementation FSMs are required to have $m \geq n$ states [14]. Other works are even more restrictive, either requiring $m = n$ [13, 15, 17] or completely specified specifications in order to generate complete test suites [7, 8, 10, 11, 17].

In a recent approach [1], necessary and sufficient conditions have been proposed for test suite completeness. The proposal assumes more relaxed constraints about the models involved in order to yield positive verdicts for test completeness. On the other hand, it still requires that candidate implementations be completely specified. Moreover, the approach

*Computing Department, University of Londrina, Londrina, Brazil, *email: bonifacio@uel.br*. Supported by FAPESP, process 2012/23500-6.

†Computing Institute, University of Campinas, Campinas, Brazil, *email: arnaldo@ic.unicamp.br*.

also relies on the classical notion of completeness, where test cases are assumed to run in both the specification and in the implementation models, even when implementations are considered as black-boxes.

In this paper, we treat more general cases in that we allow for partial models, both in the specification side and also in the implementation side. We prove that our method succeeds in determining the completeness of the test suites, even in situations where other approaches fail. We also argue by a rigorous argument that there is an upper bound on the number of states in implementations in order to still have completeness of test suites. Further, we relax the classical notion of completeness in order to deal with a more realistic situation in practice. More specifically, we allow for blocking test cases, that is, test cases which may not run to completion in neither the specification nor in the implementation machines. We also provide necessary and sufficient conditions for checking completeness of test suites in this new scenario.

The paper is organized as follows. Section 2 contains basic definitions and notations. Section 3 discusses necessary and sufficient conditions for test suite completeness for partial FSM models. In Section 4 we also prove an intrinsic relationship between the number of states in implementation candidates and the size of test cases when checking test suite completeness. In Section 5 we describe necessary and sufficient conditions for test suite completeness in the presence of blocking test cases. Section 6 concludes the paper.

2 Definitions and notation

In this section we present some definitions and notation that will be useful later. Let $\mathcal{I}$ be an alphabet. The length of any finite sequence of symbols α over $\mathcal{I}$ is indicated by $|\alpha|$. The empty sequence will be indicated by ε, with $|\varepsilon| = 0$. The set of all sequences of length k over $\mathcal{I}$ is denoted by $\mathcal{I}^k$, while $\mathcal{I}^\star$ names the set of all finite sequences from $\mathcal{I}$. When we write $\sigma = x_1 x_2 \cdots x_n \in \mathcal{I}^\star$ $(n \geq 0)$ we mean $x_i \in \mathcal{I}$ $(1 \leq i \leq n)$, unless noted otherwise. For any set C, $\mathcal{P}(C)$ denotes its power set. Given any two sets of sequences $A, B \subseteq \mathcal{I}^\star$, their symmetric difference will be indicated by $A \ominus B$, that is $A \ominus B = (\overline{A} \cap B) \cup (A \cap \overline{B})$, where $\overline{A}$ indicates the complement of A with respect to $\mathcal{I}^\star$. By $A \setminus B$ we mean set difference.

Remark 1 *$A \ominus B = \emptyset$ iff[1] $A = B$.*

Next, we write the definition of a Finite State Machine [6, 15].

Definition 1 *A FSM is a system $M = (S, s_0, \mathcal{I}, \mathcal{O}, D, \delta, \lambda)$ where*

- *S is a finite set of* states
- *$s_0 \in S$ is the initial state*
- *$\mathcal{I}$ is a finite set of* input actions *or* input events
- *$\mathcal{O}$ is a finite set of* output actions *or* output events

[1] Here, 'iff' is short for 'if and only if'.

- $D \subseteq S \times \mathcal{I}$ *is a* specification domain
- $\delta : D \to S$ *is the* transition function
- $\lambda : D \to \mathcal{O}$ *is the* output function. □

In what follows M and N will always denote the FSMs $(S, s_0, \mathcal{I}, \mathcal{O}, D, \delta, \lambda)$ and $(Q, q_0, \mathcal{I}, \mathcal{O}', D', \mu, \tau)$, respectively. Let $\sigma = x_1 x_2 \cdots x_n \in \mathcal{I}^\star$, $\omega = a_1 a_2 \cdots a_n \in \mathcal{O}^\star$ $(n \geq 0)$. If there are states $r_i \in S$ $(0 \leq i \leq n)$ such that $\delta(r_{i-1}, x_i) = r_i$ and $\lambda(r_{i-1}, x_i) = a_i$ $(1 \leq i \leq n)$, then we may write $r_0 \stackrel{\sigma/\omega}{\to} r_n$. When the input sequence σ, or the output sequence ω, is not important, then we may write $r_0 \stackrel{\sigma/}{\to} r_n$, or $r_0 \stackrel{/\omega}{\to} r_n$, respectively, and, if both sequences are not important we may write $r_0 \to r_n$. We can also drop the target state, when it is not important, *e.g.* $r_0 \stackrel{\sigma/\omega}{\to}$ or $r_0 \to$. It will be useful to extend the functions δ and λ to pairs $(s, \sigma) \in S \times \mathcal{I}^\star$. Let $\widehat{D} = \left\{ (s, \sigma) \,|\, s \stackrel{\sigma/}{\to} \right\}$. Now define the extensions $\widehat{\delta} : \widehat{D} \to S$ and $\widehat{\lambda} : \widehat{D} \to \mathcal{O}^\star$ by letting $\widehat{\delta}(s, \sigma) = r$ and $\widehat{\lambda}(s, \sigma) = \omega$ whenever $s \stackrel{\sigma/\omega}{\to} r$. When there is no reason for confusion, we may write D, δ and λ instead of $\widehat{D}$, $\widehat{\delta}$ and $\widehat{\lambda}$, respectively. Also, the function $U : S \to \mathcal{I}^\star$ will be useful, where $U(s) = \{\sigma \,|\, (s, \sigma) \in \widehat{D}\}$.

Now we are in a position to define test cases and test suites.

Definition 2 *Let M be a FSM. A* test suite for M *is any finite subset of $U(s_0)$. Any element of a test suite is a* test case. □

Since test cases must be applied from initial states, implementations under test must be brought to its initial state before the application of a test case. This can be achieved using a homing sequence [8, 17]. If there exist more than one test case to be applied, it is assumed that the implementation under test has a reset operation. The reset operation brings the machine back to its initial state [3, 4].

The notion of a simulation is given next.

Definition 3 *Let M and N be FSMs. We say that a relation $R \subseteq S \times Q$ is a* simulation *(*of M by N*) iff $(s_0, q_0) \in R$, and whenever we have $(s, q) \in R$ and $s \stackrel{x/a}{\to} r$ in M, then there is a state $p \in Q$ such that $q \stackrel{x/a}{\to} p$ in N and with $(r, p) \in R$. We say that M and N are* bi-similar *iff there are simulation relations $R_1 \subseteq S \times Q$ and $R_2 \subseteq Q \times S$.* □

3 Test suite completeness for partial FSMs

In this section we give necessary and sufficient conditions for verifying test suite completeness for FSM models. Such conditions will allow for partiality in both the specification and implementation machines.

We start by writing the classical notion of distinguishability and equivalence.

Definition 4 *Let M and N be FSMs and let $s \in S$, $q \in Q$. Let $C \subseteq \mathcal{I}^\star$. We say that s and q are C-*distinguishable *iff $\lambda(s, \sigma) \neq \tau(q, \sigma)$ for some $\sigma \in U(s) \cap U(q) \cap C$, denoted $s \not\approx_C q$. Otherwise, s and q are C-*equivalent*, denoted $s \approx_C q$. We say that M and N are C-*distinguishable *iff $s_0 \not\approx_C q_0$, and they are C-*equivalent *iff $s_0 \approx_C q_0$.* □

When C is not important, or when it is clear from the context, we might drop the index. When there is no mention to C, we understand that we are taking $C = \mathcal{I}^\star$. In this case, the condition $U(s_0) \cap U(q_0) \cap C$ reduces to $U(s_0) \cap U(q_0)$.

Next we introduce the concept of n-complete test suites.

Definition 5 *Let M be a FSM, let T a test suite for M and take $n \geq 1$. Then T is n-complete for M iff for any FSM N, with $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$, if $M \not\approx N$ then $M \not\approx_T N$.* □

The following result will help to show the existence of simulation relations.

Lemma 1 *Let M and N be FSMs. Let $n \geq 1$, $s_i \in S$, $p_i \in Q$ $(1 \leq i \leq n)$ and $x_i \in \mathcal{I}$, $a_i \in \mathcal{O}$, $b_i \in \mathcal{O}'$ $(1 \leq i < n)$ be such that $s_i \stackrel{x_i/a_i}{\rightarrow} s_{i+1}$ and $p_i \stackrel{x_i/b_i}{\rightarrow} p_{i+1}$ $(1 \leq i < n)$. Assume further that $s_1 \approx p_1$. Then $s_i \approx p_i$ $(1 \leq i \leq n)$ and $a_1 a_2 \cdots a_{n-1} = b_1 b_2 \cdots b_{n-1}$.*

Proof An easy induction on $n \geq 1$ [1]. □

The next lemma states half of our desired result. We note that specifications and candidate implementations can be partial machines.

Lemma 2 *Let T be a n-complete test suite for a FSM M. Let N be a FSM such that $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$. If $M \approx_T N$ then there exists a simulation of M by N.*

Proof Define a relation $R \subseteq S \times Q$ by letting $(s, q) \in R$ iff $\delta(s_0, \alpha) = s$ and $\mu(q_0, \alpha) = q$ for some $\alpha \in \mathcal{I}^\star$. Since $\delta(s_0, \varepsilon) = s_0$ and $\mu(q_0, \varepsilon) = q_0$ we get $(s_0, q_0) \in R$.

Now assume $(s, q) \in R$ and let $s \stackrel{x/a}{\rightarrow} r$ in M, for some $r \in S$, $x \in \mathcal{I}$ and $a \in \mathcal{O}$. Since T is n-complete for M, $U(s_0) \subseteq U(q_0)$, $|Q| \leq n$ and $M \approx_T N$, Definition 5 gives $M \approx N$. Hence, $s_0 \approx q_0$. Since $(s, q) \in R$, the construction of R gives some $\alpha \in \mathcal{I}^\star$ such that $\delta(s_0, \alpha) = s$ and $\mu(q_0, \alpha) = q$. Composing, we get $\delta(s_0, \alpha x) = \delta(s, x) = r$ and so $\alpha x \in U(s_0)$. Since $U(s_0) \subseteq U(q_0)$, we obtain $\alpha x \in U(q_0)$. Then $\mu(q_0, \alpha x) = \mu(q, x) = p$, for some $p \in Q$.

Collecting, we get $\delta(s_0, \alpha) = s$, $\mu(q_0, \alpha) = q$ and $s_0 \approx q_0$. Using Lemma 1 we conclude that $s \approx q$, and then we must have $\lambda(s, x) = a = \tau(q, x)$. So, from $(s, q) \in R$, $s \stackrel{x/a}{\rightarrow} r$ we obtained $p \in Q$ such that $q \stackrel{x/a}{\rightarrow} p$. Finally, we note that we also have $\mu(q_0, \alpha x) = p$ and $\delta(s_0, \alpha x) = r$. The definition of R now gives $(r, p) \in R$. This shows that R is a simulation relation, concluding the proof. □

We now show the converse. That is, if there is a simulation of M by any T-equivalent FSM with $U(s_0) \subseteq U(q_0)$ and with at most n states, then n-completeness of the test suite follows.

Lemma 3 *Let M be a FSM, let T a test suite for M and take $n \geq 1$. Assume that M can be simulated by any T-equivalent FSM N, with $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$. Then T is n-complete for M.*

Proof We proceed by contradiction. Assume that T is not n-complete for M. Then, by Definition 5, there exists a T-equivalent FSM N with $U(s_0) \subseteq U(q_0)$, $|Q| \leq n$, and such that $M \not\approx N$. Using Definition 4, we get an input sequence $\sigma = x_1 \dots x_n \in \mathcal{I}^\star$ $(n \geq 0)$ and an input symbol $y \in \mathcal{I}$ such that $\lambda(s_0, \alpha) = \tau(q_0, \alpha)$ and $\lambda(s_0, \alpha y) \neq \tau(q_0, \alpha y)$. Let $s_i \in S$, $q_i \in Q$ $(1 \leq i \leq n)$ be such that $\delta(s_{i-1}, x_i) = s_i$ and $\mu(q_{i-1}, x_i) = q_i$ $(1 \leq i \leq n)$. So, $\delta(s_0, \sigma) = s_n$ and $\mu(q_0, \sigma) = q_n$. Further, we get $s \in S$ and $q \in Q$ such that $\delta(s_n, y) = s$, $\mu(q_n, y) = q$ and $\lambda(s_n, y) \neq \tau(q_n, y)$.

Since N is T-equivalent to M, $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$, the hypothesis gives a simulation relation $R \subseteq S \times Q$.

Claim: $(s_i, q_i) \in R$ $(0 \leq i \leq n)$.

Proof of the Claim. We go by induction on $i \geq 0$.

Basis: we get $(s_0, q_0) \in R$ directly from Definition 3.

Induction step: assume that $(s_i, q_i) \in R$ for some $i < n$. Since $\delta(s_i, x_{i+1}) = s_{i+1}$, Definition 3 gives a $q \in Q$ such that $\mu(q_i, x_{i+1}) = q$, $\lambda(s_i, x_{i+1}) = \tau(q_i, x_{i+1})$ and $(s_{i+1}, q) \in R$. But we already have $\mu(q_i, x_{i+1}) = q_{i+1}$ and, since μ is a function, we get $q = q_{i+1}$. Thus $(s_{i+1}, q_{i+1}) \in R$ extending the induction and establishing the Claim. $\triangle$

Using the Claim, we get $(s_n, q_n) \in R$. Since $\delta(s_n, y) = s$, Definition 3 gives a $p \in Q$ such that $(s, p) \in R$, $\mu(q_n, y) = p$, and $\lambda(s_n, y) = \tau(q_n, y)$, which is a contradiction. $\square$

Putting together the previous results we obtain necessary and sufficient conditions for n-completeness of test suites, even if we allow for partial implementation candidates.

Theorem 1 *Let M be a FSM, let T be a test suite for M and let $n \geq 1$. Then, T is n-complete for M iff M can be simulated by any T-equivalent FSM N that satisfies $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$.*

Proof Assume that T is n-complete for M. Then, Lemma 2 guarantees that N can simulate M when N is T-equivalent to M, $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$. Now assume that M can be simulated by any T-equivalent FSM N such that $U(s_0) \subseteq U(q_0)$ and $|Q| \leq n$. Then, Lemma 3 guarantees that T is n-complete for M. $\square$

Remark 2 *Note that Theorem 1 is valid even in the absence of the condition $|Q| \leq n$, if Definition 5 is also changed accordingly. But removing the condition would result in a vacuous statement, as no test suite would then be complete. In Section 4 we investigate more closely the relationship between the number of states in implementations and the complete test suites.*

An algorithm for checking completeness of test suites has been investigated [1]. Given an implementation M and a test suite T, that algorithm proceeds in two steps. At the end of the first step, the algorithm constructs a number of T-equivalent candidate FSMs with at most n states, where n is an upper bound on the number of states for FSM implementations as given in Definition 5. The final step then checks if all candidate FSMs can effectively simulate M. If the answer is positive, we could declare T to be complete for M. If, on the other hand, it could be proved that any of the candidate FSMs was unable to simulate M, then T could be declared not complete for M.

Now, however, we have also to check if the condition $U(s_0) \subseteq U(q_0)$ holds, given the specification M and any candidate FSM N. If it does hold then N should be passed on to te final step that checks for a simulation relationship, otherwise it should be discarded. But this can be efficiently done by noting that from any FSM $M = (S, s_0, \mathcal{I}, \mathcal{O}, D, \delta, \lambda)$ we can readily extract a finite automaton $M_A = (S, s_0, \mathcal{I}, \delta, F)$ [16] where the set of final states is taken to be $F = S$. Let $L(M_A)$ be the language accepted by such an automaton. Then, clearly, $U(s_0) \subseteq U(q_0)$ if and only if $L(M_A) \subseteq L(N_A)$. And this test can be performed efficiently [16].

4 On the size of implementation machines

In this section we show that if one allows for too large implementations, then test completeness is lost. More specifically, if T is a test suite for a FSM M, then T is not n-complete for M, where $n > k|S|$ and k is a constant that depends only on T. This means that T may not be able to detect all faults in implementations with n or more states. We establish this result by a series of claims.

We start with some basic facts and some notation. Let $\sigma = x_0 x_1 \cdots x_k$ be a sequence of symbols over an alphabet. Then $\sigma_{i,j}$ $(0 \leq i < j \leq k+1)$ indicates the substring $x_i x_{i+1} \cdots x_{j-1}$. Let α be another sequence of symbols over the same alphabet. We say that σ is *embedded* in α if and only if there are β_i $(0 \leq i \leq k+1)$ such that $\alpha = \beta_0 x_0 \beta_1 x_1 \cdots \beta_k x_k \beta_{k+1}$. Let T be a test suite for a FSM M and let $\sigma \in T$. We say that σ is *extensible* in T if and only if $\sigma = \sigma_1 \sigma_2$ and there is some non-null γ such that $\sigma_1 \gamma \sigma_2$ is in T. Otherwise, σ is *non-extensible* in T.

We now say when a FSM is reduced.

Definition 6 *We say that a FSM $M = (S, s_0, \mathcal{I}, \mathcal{O}, D, \delta, \lambda)$ is* reduced *iff every pair of distinct states in S are distinguishable, and for all $s \in S$ there is a $\sigma \in \mathcal{I}^\star$ with $\delta(s_0, \sigma) = s$.* □

Remark 3 *Note that if M is a reduced FMS with at least two reachable, then there always exists a transition out of any reachable state s, that is $(s, x) \in D$ for some $x \in \mathcal{I}$. Otherwise, s could not be distinguished from any other reachable state in M.*

From this point on, we fix a reduced FSM M and a test suite T for M. Also, we fix $\sigma = x_0 x_1 \cdots x_k$, $k \geq 0$, as a smallest non-extensible test case in T.

Remark 4 *If $T \cap U(s_0) = \emptyset$ then any FSM is trivially T-equivalent to M. Moreover, if $\sigma = \varepsilon$, then $T = \{\varepsilon\}$ and, again, any FSM is trivially T-equivalent to M. Since M is reduced, one can easily construct a one-state FSM that is not equivalent to M. Hence, in both cases, T would not be 1-complete for M. We, therefore, can assume that such a non-null σ exists in T.*

Since $\sigma \in U(s_0)$, we get transitions $\pi_i : s_i \stackrel{x_i/a_i}{\rightarrow} s_{i+1}$ in M $(0 \leq i \leq k)$. Those are the *distinguished transitions of M*. Moreover, since M is reduced, we have $s_{k+1} \stackrel{z/a}{\rightarrow} s'$ in M, for some $z \in \mathcal{I}$, $a \in \mathcal{O}$ and $s' \in S$. We call this the *marked transition of M*.

We now construct the FSM N with the same input alphabet, $\mathcal{I}$, and the same output alphabet, $\mathcal{O}$. Let $Q = S \times [0, k+1]$, that is, the states of N are pairs $[q, i]$ where q is a state of M and $0 \leq i \leq k+1$. The initial state of N is $q_0 = [s_0, 0]$. We complete the specification of N by listing its transitions:

(a) If $s \xrightarrow{y/b} r$ is not a distinguished transition of M, let $[s, i] \xrightarrow{y/b} [r, i]$ be a transition in N, for all i, $0 \leq i \leq k$.

(b) For all distinguished transitions $s_i \xrightarrow{x_i/a_i} s_{i+1}$ of M, let $[s_i, i] \xrightarrow{x_i/a_i} [s_{i+1}, i+1]$ be a transition in N. We call these the *distinguished transitions of* N.

(c) If $s \xrightarrow{y/b} r$ is not the marked transition of M, we let $[s, k+1] \xrightarrow{y/b} [r, k+1]$ be a transition in N.

(d) For the marked transition of M, $s_{k+1} \xrightarrow{z/a} s'$, we let $[s_{k+1}, k+1] \xrightarrow{z/b} [s', k+1]$, for some $b \neq a$, be a transition in N.

This completes the specification of N. Easily, N has $(|\sigma| + 1)|S|$ states.

The next facts make explicit the behavior of the construction.

Fact 1 *Let $\pi : s \xrightarrow{\alpha/\omega} p$ in M and take $0 \leq i \leq k+1$. Then in N we must have $[s, i] \xrightarrow{\alpha/\omega'} [p, j]$ for some $j \geq i$. Moreover, $\omega = \omega'$ if the marked transition of M does not occur in π.*

Proof By induction on $|\alpha| = n \geq 0$. When $n = 0$ the result follows immediately.

For the induction step, let $\alpha = \beta x$, $\omega = \rho a$, with $x \in \mathcal{I}$, $a \in \mathcal{O}$, and $\pi : s \xrightarrow{\beta/\rho} r \xrightarrow{x/a} p$. The induction hypothesis gives $\pi_1 : [s, i] \xrightarrow{\beta/\rho'} [r, j]$ in N, with $j \geq i$.

If $j = k+1$, then items (c) and (d) in the construction of N give $[r, j] \xrightarrow{x/a'} [p, j]$ in N. Then, clearly, $[s, i] \xrightarrow{\alpha/\omega'} [p, j]$ in N, where $\omega' = \rho' a'$. Moreover, if the marked transition of M does not occur in π then the induction hypothesis gives $\rho = \rho'$. Also, since $r \xrightarrow{x/a} p$ is not the marked transition of M, item (c) of the construction of N yields $a' = a$. We conclude that $\omega = \rho a = \rho' a' = \omega'$, as desired.

Now take $j < k+1$. Then items (a) and (b) of the construction give $[r, j] \xrightarrow{x/a'} [p, \ell]$ in N where $\ell = j$ or $\ell = j+1$. Hence, $[s, i] \xrightarrow{\alpha/\omega'} [p, j]$ with $\omega' = \rho' a'$ and, in any case, $\ell \geq j \geq i$, as desired. Again, if the marked transition of M does not occur in α then we get $\rho = \rho'$ using the induction hypothesis. Clearly, in items (a) and (b) we have $a' = a$. This readily gives $\omega = \rho a = \rho' a' = \omega'$, concluding the proof. □

The next result gives the converse.

Fact 2 *Let $\pi : [s, i] \xrightarrow{\alpha/\omega} [p, j]$ in N. Then, we have (i) $j \geq i$, (ii) $\sigma_{i,j}$ is embedded in α, and (iii) $s \xrightarrow{\alpha/\omega'} p$ in M. Moreover, $\omega = \omega'$ if the marked transition of N does not occur in π.*

Proof By induction on $|\alpha| = n \geq 0$. When $n = 0$ the result follows easily.

For the induction step, let $\alpha = \beta x$, $\omega = \rho a$, with $x \in \mathcal{I}$, $a \in \mathcal{O}$, and $\pi' : [s, i] \stackrel{\beta/\rho}{\to} [r, \ell] \stackrel{x/a}{\to} [p, j]$. The induction hypothesis gives $\ell \geq i$, $\sigma_{i,\ell}$ embedded in β, and $s \stackrel{\beta/\rho'}{\to} r$ in M. According to the items in the construction of N we have four cases for the transition $[r, \ell] \stackrel{x/a}{\to} [p, j]$:

(a) It was added because of item (a). Then, $\ell = j$ and $r \stackrel{x/a}{\to} p$ is in M. We get $j = \ell \geq i$ and $\sigma_{i,j} = \sigma_{i,\ell}$ is embedded in α, as desired. Composing we get $s \stackrel{\beta x/\omega'}{\to} p$ in M, with $\beta x = \alpha$ and $\rho' a = \omega'$. If the marked transition of M does not occur in π, then $\rho = \rho'$ by the induction hypothesis. So, $\omega = \rho a = \rho' a = \omega'$, as we wanted.

(b) It was added because of item (b). Then, $x = x_\ell$, $j = \ell + 1$, and $r \stackrel{x/a}{\to} p$ in M. Clearly, (i) and (iii) hold, with $\omega' = \rho' a$. Also, $\sigma_{i,j} = \sigma_{i,\ell+1} = \sigma_{i,\ell} x_\ell$. Since $\alpha = \beta x = \beta x_\ell$ and $\sigma_{i,\ell}$ embedded in β, we conclude that $\sigma_{i,j}$ is embedded in α. If the marked transition of M does not occur in π, then we proceed as in case (a) and obtain $\omega = \rho a = \rho' a = \omega'$, as needed.

(c) It was added because of item (c). Now we have $\ell = k + 1 = j$ and $r \stackrel{x/a}{\to}$ in M, showing that (i) and (iii) hold with $s \stackrel{\beta x/\omega'}{\to} p$ and $\omega' = \rho' a$. We have that $\sigma_{i,\ell} = \sigma_{i,j}$ is already embedded in β and so its also embedded in α, given that $\alpha = \beta x$. The reasoning to obtain $\omega = \omega'$ is the same as in case (a).

(d) It was added because of item (d). Proceed exactly as in case (c). Now, the marked transition of N does occur in π and so the last statement of the Fact holds vacuously. This last case concludes the proof. □

The last two results already establish that the same sequences of input symbols can be run in both machines.

Fact 3 $U(s_0) = U(q_0)$.

Proof Recall that $q_0 = [s_0, 0]$. Let $s_0 \stackrel{\alpha/}{\to}$ in M. Using Fact 1 we get $[s_0, 0] \stackrel{\alpha/}{\to}$ in N. Hence, $U(s_0) \subseteq U(q_0)$. In a similar way we can get $U(q_0) \subseteq U(s_0)$ using Fact 2, and the result follows. □

We are now in a position to show that M and N are T-equivalent.

Fact 4 $M \approx_T N$.

Proof We go by contradiction. Assume we have $\alpha x \in T \cap U(s_0) \cap U(q_0)$, $x \in \mathcal{I}$ such that $s_0 \stackrel{\alpha/\omega}{\to} s \stackrel{x/a}{\to} r$ in M and $[s_0, 0] \stackrel{\alpha/\omega}{\to} [q, i] \stackrel{x/b}{\to} [p, j]$ in N, with $a \neq b$. Fact 2 gives $s_0 \stackrel{\alpha/}{\to} q$ in M. But we already have $s_0 \stackrel{\alpha/}{\to} s$ in M, and so we conclude that $s = q$. Using Fact 2 again, from $s \stackrel{x/}{\to} r$ in M and $[s, i] \stackrel{x/}{\to} [p, j]$ in N we get $p = r$. We can now write $\pi : [s, i] \stackrel{x/b}{\to} [r, j]$ in N

and $s \stackrel{x/a}{\rightarrow} r$ in M with $a \neq b$. From the construction of N we conclude that π is the marked transition of N. Hence, $i = j = k+1$. We now have $[s_0, 0] \stackrel{\alpha/\omega}{\rightarrow} [s, k+1]$ in N. From Fact 2, $\sigma = \sigma_{0,k+1}$ is embedded in α and so σ is embedded in αx. Since $\alpha x \in T$, we conclude that σ is extensible in T. But this contradicts the choice of σ, completing the proof. □

In the opposite direction, the next result shows that M and N are not equivalent.

Fact 5 *$M \not\approx N$.*

Proof Since $\sigma \in U(s_0)$, Fact 3 gives $\sigma \in U(q_0)$. By the choice of σ, in M we have $s_0 \stackrel{\sigma/\omega}{\rightarrow} s_{k+1}$. Further, by the choice of z and a, we have $s_{k+1} \stackrel{z/a}{\rightarrow} s'$ in M. Hence, $s_0 \stackrel{\sigma z/\omega a}{\rightarrow} s'$ in M. Item (b) of the construction of N gives $[s_i, i] \stackrel{x_i/a_i}{\rightarrow} [s_{i+1}, i+1]$, $0 \leq i \leq k$. Then, $[s_0, 0] \stackrel{\sigma/\omega}{\rightarrow} [s_{k+1}, k+1]$ in N. By item (d) of the construction of N we get $[s_{k+1}, k+1] \stackrel{z/b}{\rightarrow} [s', k+1]$ in N. Composing, we obtain $[s_0, 0] \stackrel{\sigma z/\omega b}{\rightarrow} [s', k+1]$ in N. This shows that $M \not\approx N$, because $a \neq b$. □

Collecting, we can show that a test suite T will not be n-complete for a FSM M when n is larger than a certain bound, which depends only on M and T.

Theorem 2 *Let M be a FSM and let T be a test suite for M. Let σ be a shortest test case in T that is non-extensible in T. Then T is not $\big((|\sigma|+1)|S|\big)$-complete for M.*

Proof The construction of N yields a machine that is T-equivalent to M, using Fact 4. We also know that M and N are not equivalent, by Fact 5. Also, using Fact 3, we know that $U(s_0) \subseteq U(q_0)$. Since N has $n = (|\sigma|+1) \times |S|$ states, Definition 5 says that T is not n-complete for M. □

Next, we give a simple example to illustrate the construction of machine N. Let $M = (S, s_0, \mathcal{I}, \mathcal{O}, D, \delta, \lambda)$ be a specification FSM as depicted in Figure 1. The set of states is $S = \{s_0, s_1\}$, $\mathcal{I} = \mathcal{O} = \{0, 1\}$, and D, δ, λ are given as depicted in the figure. Note that M is a partial FSM since $(s_1, 1) \notin D$. Also let $T = \{0000, 100\}$ be a test suite for M. We notice

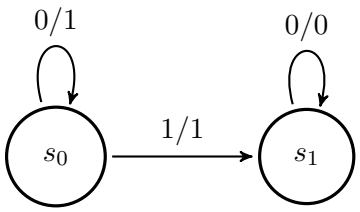

Figure 1: FSM specification M.

that T is 2-complete for M, *i.e.*, for implementation FSMs with at most as many states as M. This can be checked by using the algorithm described in Section 3.

Now take $\sigma = 100$ as the shortest test case in T that is non-extensible in T. Then we construct a candidate implementation N by applying items (a) to (d) with $(|\sigma|+1)|S| =$

$(3+1)2 = 8$ states. From item (a) we create transitions $[s_0, i] \stackrel{0/1}{\to} [s_0, i]$, for all i, $0 \leq i \leq 2$. We also obtain the distinguished transitions $[s_0, 0] \stackrel{1/1}{\to} [s_1, 1]$, $[s_1, 1] \stackrel{0/0}{\to} [s_1, 2]$, $[s_1, 2] \stackrel{0/0}{\to} [s_1, 3]$ $[s_0, 1] \stackrel{1/1}{\to} [s_1, 2]$, $[s_0, 2] \stackrel{1/1}{\to} [s_1, 3]$ and $[s_1, 0] \stackrel{0/0}{\to} [s_1, 1]$ by using item (b). From item (c) we get the transitions $[s_0, 3] \stackrel{0/1}{\to} [s_0, 3]$, $[s_0, 3] \stackrel{0/1}{\to} [s_0, 3]$ and $[s_0, 3] \stackrel{1/1}{\to} [s_1, 3]$. Finally we complete the machine N with the marked transition $[s_3, 3] \stackrel{0/1}{\to} [s_3, 3]$ as given by item (d). The machine N is depicted in Figure 2. However it is a simple matter to see that states $[s_0, 1]$, $[s_0, 2]$, $[s_0, 3]$ and $[s_1, 0]$ are not reachable in N. Then we can remove such transitions in order to obtain a reduced FSM as depicted in Figure 3. Note that we rename the remaining

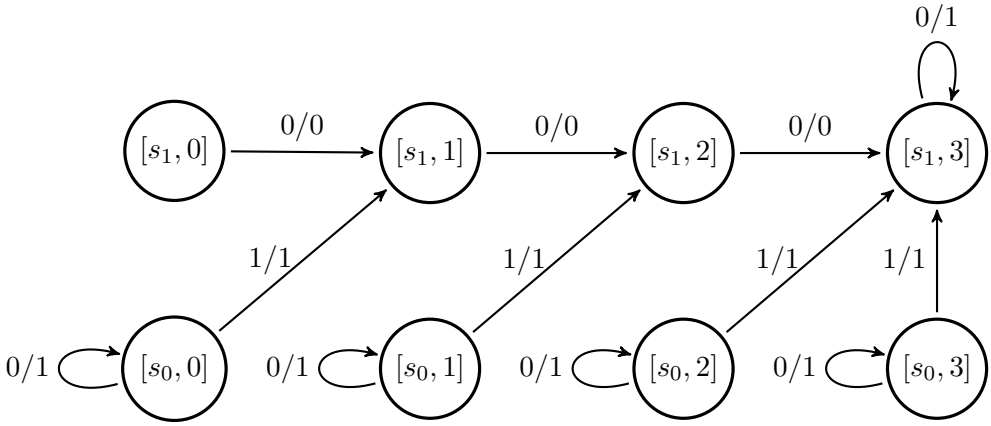

Figure 2: A candidate implementation N.

transitions by letting $q_0 = [s_0, 0]$, $q_1 = [s_1, 1]$, $q_2 = [s_1, 2]$, and $q_3 = [s_1, 3]$.

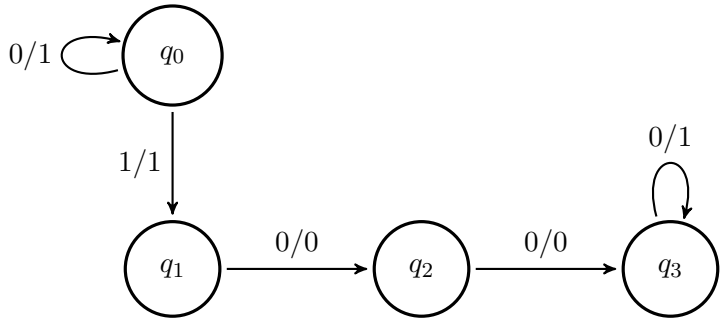

Figure 3: A reduced candidate implementation N.

Now we can easily check that $M \approx_T N$ because $\lambda(s_0, 0000) = 1111 = \tau(q_0, 0000)$ and $\lambda(s_0, 100) = 100 = \tau(q_0, 100)$, but $M \not\approx N$ since we have $\lambda(s_0, 1000) = 1000 \neq 1001 = \tau(q_0, 1000)$. It is also easy to verify that $U(s_0) \subseteq U(q_0)$. Hence we conclude that T is complete for M taking into account machines with more than eight states. In this specific case, where $T = \{0000, 100\}$, we show that T is not complete for M through a candidate implementation N with four states. In fact, those states that are not reachable in the

machine N can be removed from it. Therefore it is sufficient to construct an implementation candidate N as depicted in Figure 3 to show that T is not complete for M when considering implementations with more than four states. We noticed that, in general, for any FSM specification M we have that T is not $(|\sigma|+1)|S|$-complete for M, where $\sigma \in T$ is a shortest non-extensible test case in T, as proved in Theorem 2.

5 Completeness in the presence of blocking test cases

In this section we allow for test cases that may not run to completion in candidate implementations. That is, when put under test, implementations may not output the same number of events as there were input symbols in the test case. That this kind of fault can be readily identified by observing the external behavior of the model.

Given a FSM model M, a *blocking* test case for M is one that does not run to completion in M. Given a test suite T, two FSM models M and N will be considered as T-equivalent in the presence of blocking test cases, if all blocking test cases for M in T are also blocking for N, and vice-versa. Furthermore, any test case that is non-blocking for both M and N must output identical behaviors when run through both models. We then investigate necessary and sufficient conditions for T to be a complete test suite, when considering this more general scenario.

We start by making precise the new notion of equivalent models.

Definition 7 *Let M and N be FSMs and let $s \in S$, $q \in Q$. Let $C \subseteq \mathcal{I}^\star$. We say that s and q are C-*alike*, denoted $s \sim_C q$, iff $\big(U(s) \ominus U(q)\big) \cap C = \emptyset$ and $\lambda(s,\sigma) = \tau(q,\sigma)$ for all $\sigma \in U(s) \cap U(q) \cap C$. Otherwise, s and q are C-*unlike*, denoted $s \not\sim_C q$. We say that M and N are C-*alike *iff $s_0 \sim_C q_0$, otherwise they are C-*unlike*.* □

Again, when C is not important, or when it is clear from the context, we might drop the index, and when there is no mention to C, we understand that we are taking $C = \mathcal{I}^\star$.

Remark 5 *Using Remark 1, we note that $s \sim q$ is equivalent to $U(s) = U(q)$ and $\lambda(s,\sigma) = \tau(q,\sigma)$ for all $\sigma \in U(s)$.*

The new notion of test suite completeness now reflects the fact that we may be in the presence of blocking test cases. In order to avoid ambiguities we rename completeness to perfectness.

Definition 8 *Let M be a FSM and T be a test suite for M. Then T is* perfect for M *iff for any FSM N, if $M \not\sim N$ then $M \not\sim_T N$.* □

That is when T is a perfect test suite for a specification M, then for any implementation under test N, if M and N are unlike, then they are also T-unlike.

The following result will be useful to certain bi-similarities.

Lemma 4 *Let M and N be FSMs. Let $n \geq 1$, $s_i \in S$, $p_i \in Q$ $(1 \leq i \leq n)$ and $x_i \in \mathcal{I}$, $a_i \in \mathcal{O}$, $b_i \in \mathcal{O}'$ $(1 \leq i < n)$ be such that $s_i \xrightarrow{x_i/a_i} s_{i+1}$ and $p_i \xrightarrow{x_i/b_i} p_{i+1}$ $(1 \leq i < n)$. Assume further that $s_1 \sim p_1$. Then $s_i \sim p_i$ $(1 \leq i \leq n)$ and $a_1 a_2 \cdots a_{n-1} = b_1 b_2 \cdots b_{n-1}$.*

Proof Let $\sigma = x_1x_2\cdots x_{n-1}$, $\omega_1 = a_1a_2\cdots a_{n-1}$ and $\omega_2 = b_1b_2\cdots b_{n-1}$. We clearly have $s_1 \stackrel{\sigma/\omega_1}{\rightarrow} s_n$ and $p_1 \stackrel{\sigma/\omega_2}{\rightarrow} p_n$. Definition 7 immediately gives $\omega_1 = \omega_2$, because $s_1 \sim p_1$ and $\sigma \in U(s_1) \cap U(q_1)$.

To see that $s_i \sim p_i$ $(1 \leq i \leq n)$ we go by induction on n. The basis is trivial and we proceed with the induction step. Let $1 \leq k < n$ and assume $s_k \sim p_k$. Let $\alpha = x_1 \cdots x_k$. Clearly $\delta(s_1, \alpha) = s_{k+1}$, $\mu(p_1, \alpha) = p_{k+1}$ and so $\alpha \in U(s_1) \cap U(p_1)$. For te sake of contradiction, assume that $s_{k+1} \not\sim p_{k+1}$. By Definition 7 we have two cases.

Case 1: $U(s_{k+1}) \ominus U(p_{k+1}) \neq \emptyset$.

Let $\beta \in U(s_{k+1})$ and $\beta \notin U(p_{k+1})$. This gives $\alpha\beta \in U(s_1)$ and $\alpha\beta \notin U(p_1)$. Hence $U(s_1) \ominus U(p_1) \neq \emptyset$, contradicting $s_1 \sim p_1$. The situation $\beta \notin U(s_{k+1})$ and $\beta \in U(p_{k+1})$ is entirely analogous.

Case 2: $\beta \in U(s_{k+1}) \cap U(p_{k+1})$ and $\lambda(s_{k+1}, \beta) \neq \tau(p_{k+1}, \beta)$, for some $\beta \in \mathcal{I}^\star$.

This gives $\alpha\beta \in U(s_1) \cap U(p_1)$. Moreover,

$$\lambda(s_1, \alpha\beta) = \lambda(s_1, \alpha)\lambda(\delta(s_1, \alpha), \beta)) = \lambda(s_1, \alpha)\lambda(s_{k+1}, \beta), \text{ and}$$
$$\tau(p_1, \alpha\beta) = \tau(p_1, \alpha)\tau(\mu(p_1, \alpha), \beta)) = \tau(p_1, \alpha)\tau(p_{k+1}, \beta).$$

Because $|\lambda(s_1, \alpha)| = |\tau(p_1, \alpha)|$ and $\lambda(s_{k+1}, \beta) \neq \tau(p_{k+1}, \beta)$, we get $\lambda(s_1, \alpha\beta) \neq \tau(p_1, \alpha\beta)$. Since $\alpha\beta \in U(s_1) \cap U(p_1)$, this contradicts $s_1 \sim p_1$.

The proof is complete. □

The next result guarantees the existence of bi-simulations in the presence of blocking test cases.

Lemma 5 *Let T be a perfect test suite for a FSM M. Let N be a FSM such that $M \sim_T N$. Then M and N are bi-similar.*

Proof Define a relation $R_1 \subseteq S \times Q$ by letting $(s, q) \in R_1$ if and only if $\delta(s_0, \alpha) = s$ and $\mu(q_0, \alpha) = q$ for some $\alpha \in \mathcal{I}^\star$, $s \in S$ and $q \in Q$. Since $\delta(s_0, \varepsilon) = s_0$ and $\mu(q_0, \varepsilon) = q_0$ we get $(s_0, q_0) \in R_1$.

Now assume $(s, q) \in R_1$ and let $s \stackrel{x/a}{\rightarrow} r$ for some $r \in S$, $x \in \mathcal{I}$ and $a \in \mathcal{O}$. Since $(s, q) \in R_1$, the definition of R gives some $\alpha \in \mathcal{I}^\star$ such that $\delta(s_0, \alpha) = s$ and $\mu(q_0, \alpha) = q$. Composing, we get $\delta(s_0, \alpha x) = \delta(s, x) = r$ and so $\alpha x \in U(s_0)$. Since T is perfect for M and $M \sim_T N$, Definition 7 gives $M \sim N$, that is $s_0 \sim q_0$. Further, Definition 7 and Remark 5 imply $U(s_0) = U(q_0)$, and so $\alpha x \in U(q_0)$. Then $\mu(q, x) = p$, for some $p \in Q$. Since $s_0 \sim q_0$, $\delta(s_0, \alpha) = s$ and $\mu(q_0, \alpha) = q$, Lemma 4 gives $s \sim q$. But $x \in U(s) \cap U(q)$, and so we must have $a = \lambda(s, x) = \tau(q, x)$. Thus, we have found $p \in Q$ with $q \stackrel{x/a}{\rightarrow} p$. Since $\delta(s_0, \alpha x) = r$ and $\mu(q_0, \alpha x) = p$, we also have $(r, p) \in R_1$. This shows that R_1 is a simulation relation.

A similar argument will show that $R_2 \subseteq Q \times S$, where $R_2 = R_1^{-1}$, is also a simulation relation. Thus M and N are bi-similar, as desired. □

We now show the converse, that is, if M is bi-similar to any FSM N that is T-alike to it, then T is a perfect test suite for M.

Lemma 6 *Let M be a FSM and T a test suite for M. Assume that any FSM that is T-alike to M is bi-similar to it. Then T is perfect for M.*

Proof We proceed by contradiction. Assume that T is not perfect for M. Then, by Definition 8, there exists a FSM N such that $M \sim_T N$ and $M \not\sim N$. Hence, since $M \sim_T N$, we get that N is bi-similar to M, and so there are simulation relations $R_1 \subseteq S \times Q$ and $R_2 \subseteq Q \times S$.
Claim: Let $s = \delta(s_0, \alpha)$ and $q = \mu(q_0, \alpha)$ for some $\alpha \in \mathcal{I}^\star$. Then $(s, q) \in R_1$ and $(q, s) \in R_2$.
Proof of the Claim. An easy induction on $|\alpha| \geq 0$.

Since $M \not\sim N$, by Definition 7 we have two cases:

Case 1: $\alpha \in U(s_0) \ominus U(q_0)$, for some $\alpha \in T$.

We may assume that $|\alpha|$ is minimum.

If $\alpha \in U(q_0)$ and $\alpha \notin U(s_0)$, then we may write $\alpha = \beta x$, where $\beta \in \mathcal{I}^\star$, $x \in \mathcal{I}$ are such that $\beta \in U(q_0) \cap U(s_0)$. Thus, $\delta(s_0, \beta) = s$, $\mu(q_0, \beta) = q$ and $\mu(q, x) = p$, for some $s \in S$ and some $q, p \in Q$. Since $(q_0, s_0) \in R_2$, we can use Lemma 4 and write $(q, s) \in R_2$. Because R_2 is a simulation and $\mu(q, x) = p$ we get some $r \in S$ such that $\delta(s, x) = r$. But this gives $\delta(s_0, \alpha) = \delta(s_0, \beta x) = \delta(s, x) = r$, that is $\alpha \in U(s_0)$, a contradiction.

When $\alpha \notin U(q_0)$ and $\alpha \in U(s_0)$, the argument is analogous.

Case 2: There is some $\alpha \in U(s_0) \cap U(q_0)$ with $\lambda(s_0, \alpha) \neq \tau(q_0, \alpha)$.

Again, assume that $|\alpha|$ is minimum. Then, there are $\beta \in \mathcal{I}^\star$, $x \in \mathcal{I}$, $s \in S$ and $q \in Q$ such that $\alpha = \beta x$ and $\delta(s_0, \beta) = s$, $\mu(q_0, \beta) = q$. Further, we get some $r \in S$, $p \in Q$ such that $\delta(s, x) = r$, $\mu(q, x) = p$, $a = \lambda(s, x) \neq \tau(q, x) = b$. Using the Lemma 4, we may write $(s, q) \in R_1$. Because we have $s \xrightarrow{x/a} r$ in M and R_1 is a simulation, we know that there is some $t \in Q$ such that $q \xrightarrow{x/a} t$ in N, with $(r, t) \in R_1$. But we already had $q \xrightarrow{x/b} p$ in N. Hence, since τ is a function, we conclude that $a = b$, which is a contradiction.

The proof is now complete. □

Combining the previous results we obtain necessary and sufficient conditions for the perfectness of test suites.

Theorem 3 *Let M be a FSM and T be a test suite for M. Then T is perfect for M iff any T-alike FSM is bi-similar to M.*

Proof Assume that T is perfect for M. Lemma 5 guarantees that N and M are bi-similar when N is T-alike to M. Now assume that any T-alike FSM is bi-similar to M. In this case, Lemma 6 guarantees that T is perfect for M. □

6 Conclusions

In this work we showed necessary and sufficient conditions for checking completeness of test suites for more relaxed FSM models. Our approach is general in the sense that specifications and implementations models are required to be only deterministic. So partial FSM machines are treated. We also gave an intrinsic relationship between the upper bound on the number of states in implementation candidates and the test cases of a test suite when checking completeness.

Further we described a new approach for checking completeness of test suites taking into account FSM models that allow for blocking test cases. For that, we introduced the new notion of test suite perfectness, a relaxation of the classical notion of test suite completeness. We have also provided necessary and sufficient conditions for checking test suite perfectness.

All claims herein are proved correct by rigorous arguments. Being based on necessary and sufficient proofs, our approaches always provide definitive answers regarding test suite completeness and perfectness, thus never issuing inconclusive verdicts.

We leave for future works the possibility of testing our algorithms in practical situations, and comparing the results with other methods for testing test suites completeness and perfectness.

References

[1] Adilson Luiz Bonifacio and Arnaldo Vieira Moura. Necessity and sufficiency for checking m-completeness of test suites. Technical Report IC-13-21, Institute of Computing, University of Campinas, September 2013. http://www.ic.unicamp.br/∼reltech/2013/13-21.pdf.

[2] Adilson Luiz Bonifacio, Arnaldo Vieira Moura, and Adenilso da Silva Simão. Model partitions and compact test case suites. *Int. J. Found. Comput. Sci.*, 23(1):147–172, 2012.

[3] T. S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, 1978.

[4] Rita Dorofeeva, Khaled El-Fakih, and Nina Yevtushenko. An improved conformance testing method. In *FORTE*, pages 204–218, 2005.

[5] S. Fujiwara, G. V. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, June 1991.

[6] A. Gill. *Introduction to the theory of finite-state machines.* McGraw-Hill, New York, 1962.

[7] G. Gonenc. A method for the design of fault detection experiments. *IEEE Trans. Comput.*, 19(6):551–558, 1970.

[8] F. C. Hennie. Fault detecting experiments for sequential circuits. In *Proceedings of the Fifth Annual Symposium on Switching Circuit Theory and Logical Design, 11-13 November 1964, Princeton, New Jersey, USA*, pages 95–110. IEEE, 1964.

[9] R. M. Hierons. Separating sequence overlap for automated test sequence generation. *Automated Software Engg.*, 13(2):283–301, 2006.

[10] Robert M. Hierons and Hasan Ural. Reduced length checking sequences. *IEEE Trans. Comput.*, 51(9):1111–1117, September 2002.

[11] Robert M. Hierons and Hasan Ural. Optimizing the length of checking sequences. *IEEE Trans. Comput.*, 55(5):618–629, May 2006.

[12] A. Petrenko and G. V. Bochmann. On fault coverage of tests for finite state specifications. *Computer Networks and ISDN Systems*, 29:81–106, 1996.

[13] Alex Petrenko and Nina Yevtushenko. On test derivation from partial specifications. In *In FORTE*, pages 85–102, 2000.

[14] Alexandre Petrenko and Nina Yevtushenko. Testing from partial deterministic fsm specifications. *IEEE Trans. Comput.*, 54(9):1154–1165, September 2005.

[15] Adenilso da Silva Simao and Petrenko Petrenko. Checking completeness of tests for finite state machines. *IEEE Trans. Computers*, 59(8):1023–1032, 2010.

[16] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.

[17] Hasan Ural, Xiaolin Wu, and Fan Zhang. On minimizing the lengths of checking sequences. *IEEE Trans. Comput.*, 46(1):93–99, January 1997.

[18] Ming Yu Yao, Alexandre Petrenko, and Gregor von Bochmann. Fault coverage analysis in respect to an fsm specification. In *INFOCOM*, pages 768–775, 1994.