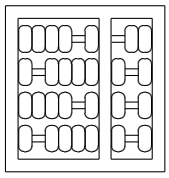

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Intrinsic Properties of Complete Test Suites

Adilson Luiz Bonifacio *Arnaldo Vieira Moura*

Technical Report - IC-14-15 - Relatório Técnico

September - 2014 - Setembro

Intrinsic Properties of Complete Test Suites

Adilson Luiz Bonifacio* Arnaldo Vieira Moura†

Abstract

One desirable property of test suites is completeness. Roughly, completeness guarantees that non-equivalent implementations under test will always be identified. Several approaches proposed sufficient, and sometimes also necessary, conditions on the specification model and on the test suite in order to guarantee completeness. Usually, these approaches impose several restrictions on the specification and on the implementations, such as requiring them to be reduced or complete. Further, test cases are required to be non-blocking on both the specification and the implementation. In this work we deal with the more general scenario where test cases can be blocking. We propose a new notion of equivalence, define a new notion that captures completeness, and move to characterize test suite completeness in this new scenario. A related issue that concerns test suite completeness is the size of implementations under test. Usually, earlier works constrain implementations to have at most the same number of states as the given specification. We establish an upper bound on the number of states of the implementations beyond which the test suite will not be complete, in the classical sense.

1 Introduction

Completeness of test suites has been studied for formal models based on Finite State Machines (FSMs) [BMdSS12, HU02, DEFY05, SP10, BM14a, UWZ97, SPY12]. A test suite with full fault detection is called complete for a FSM specification when it provides complete fault coverage [BMdSS12, HU02].

Several works have proposed strategies for generating complete test suites [dSSPY09], or for checking if a given test suite is complete for a given specification [BM14a]. Some of them presented necessary conditions [PB96, YPvB94] for test suite completeness, whereas other approaches gave sufficient, but not necessary, conditions for test suite completeness [DEFY05, PY00, SP10, UWZ97]. Some more recent works have described necessary *and* sufficient conditions for test suite completeness [BM14a, dSSPY09]. All these works imposed restrictions on the specification and implementations models, or over the fault domains [DEFY05, PY00, SP10, UWZ97, BM14a]. Some of them considered specifications with n states and restricted the implementations under test to have at most n states. Further, in some approaches specification and implementation models are required to be

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

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

reduced or completely specified machines. Always, test cases have been required to be non-blocking on both the specifications and the implementations.

In this work we deal with the more general scenario where test cases can be blocking, *i.e.*, some test cases in the test suite might not run to completion in the specification or in the implementation under test. We propose a new notion of equivalence, called alikeness, and we extend the classical notion of equivalence when blocking test cases can be present, thus giving rise to the notion of perfectness. We then use bi-simulation relations to characterize test suite perfectness in this new more general scenario.

A related issue that concerns test suite completeness is the maximum size of implementations that can be put under test. Usually, earlier works constrain implementations to have at most the same number of states as the given specification. We establish an upper bound on the number of states of implementations under test, beyond which no test suite can be complete, in the classical sense. The bound is based on the test suite size and the number of states in the given specification.

We organize the paper as follows. Basic definitions and notations appear in Section 2. In Section 3 we characterize test completeness in the presence of blocking test cases, and discuss a small illustrative example. In Section 4 we establish an upper bound on the number of states in candidate implementations beyond which no test suite is complete. Section 5 states some conclusions.

2 Definitions and notation

Let $\mathcal{I}$ be an alphabet. The length of any finite sequence α of symbols 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. 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 $\mathcal{I}^\star$. The usual set difference is indicated by $A \setminus B$.

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

Next, we write the definition of a Finite State Machine [BM14a, Gil62].

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
- *$D \subseteq S \times \mathcal{I}$ is a* specification domain

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

- $\delta : D \to S$ *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 when both sequences are not important we may write $r_0 \to r_n$. We can also drop the target state, and write $r_0 \stackrel{\sigma/\omega}{\to}$ 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 $\widehat{D} = \left\{ (s, \sigma) \,|\, s \stackrel{\sigma/}{\to} \right\}$. 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 $\mathcal{I}^\star$. Any element of a test suite is a* test case. □

Before we can define test completeness, we need the classical notions of distinguishability and equivalence.

Definition 3 *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 $U(s_0) \cap U(q_0)$. For the ease of notation, we also write $M \approx_C N$ when M and N are C-equivalent, and $M \not\approx_C N$ when they are C-distinguishable.

Now we can state the conventional notion of a n-complete test suite.

Definition 4 *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 $M \not\approx_T N$.* □

Now we say when FSM models are reduced.

Definition 5 *Let M be a FSM. We say that M is* reduced *if every pair of distinct states in S are distinguishable.* □

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 are considered T-equivalent in the

presence of blocking test cases when 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. In this case M and N will be said to be *alike*.

Definition 6 *Let M and N be reduced 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*.* □

We may also write $M \sim_C N$ when M and N are C-alike, or $M \not\sim_C N$ when 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 2 *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 notion of *perfectness* has been introduced by Bonifacio and Moura [BM14b, BM13], in order to cope with test cases that may not run to completion in specification and implementation models. Here, we redefined this notion by considering only reduced machines. From now on, when treating blocking test cases and perfectness we will always use FSM models that are reduced.

Definition 7 *Let M be a reduced FSM and T be a test suite for M. Then T is* perfect for *M iff for any reduced FSM N, if $M \not\sim N$ then $M \not\sim_T N$.* □

That is if 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. In [BM14b, BM13] bi-simulation was used to characterize test suite perfectness.

For completeness, simulations and bi-simulations are redefined next.

Definition 8 *Let M and N be reduced 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 $s \xrightarrow{x/a} r$ in M, then there is a state $p \in Q$ such that $q \xrightarrow{x/a} p$ 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 $R_2 \subseteq Q \times S$.* □

The next theorem is from [BM14b, BM13], but now restricted to reduced FSM models.

Theorem 1 *Let M be a reduced FSM and T be a test suite for M. Then T is perfect for M iff any reduced T-alike FSM is bi-similar to M.*

It is a simple matter to verify that the proofs in [BM14b, BM13] also stand when we are treating only reduced FSM models.

3 Perfectness and Isomorphism

In this section we characterize perfectness in terms of isomorphisms between FSMs. Two FSMs are said to be isomorphic when they specify exactly the same model, except for a state relabeling.

Definition 9 *Let M and N be FSMs with $\mathcal{O} = \mathcal{O}'$. An* isomorphism *(of M into N) is a bijection $f : S \to Q$ such that*

1. *$f(s_0) = q_0$; and*
2. *$s \stackrel{x/a}{\to} r$ in M if and only if $f(s) \stackrel{x/a}{\to} f(r)$ in N, for all $x \in \mathcal{I}$, $a \in \mathcal{O}$.*

Machines M and N are isomorphic *iff there is an isomorphism of M into N.* □

Remark 3 *Let M and N be FSMs. The following are immediate consequences:*

1. *f is an isomorphism of M into N if and only if f^{-1} is an isomorphism of N into M.*
2. *Any isomorphism of M into N is also a simulation of M by N.*

The first half of the characterization is easily obtained.

Lemma 2 *Let M and N be isomorphic FSMs. Then, M and N are bi-similar.*

Proof Using Remark 3, we have a simulation of M by N and vice-versa. □

Now let M and N be bi-similar. It is clear that if all states in M are distinguishable, but N has two distinct states that are equivalent, then it is possible for M and N not to be isomorphic, since these two distinct equivalent states in N would have to correspond to a single state in M. Machines illustrated in Figures 1 and 2 are a case in point. We prevent

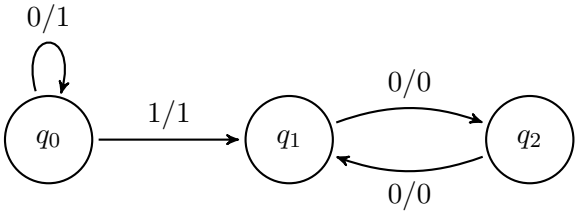

Figure 1: FSM N_1.

this situation by requiring that both M and N be reduced FSMs.

We proceed to show, by a series of claims, that if M and N are bi-similar and reduced, then they are isomorphic. We start by noting that the bi-similarity condition gives two simulation relations $R_1 \subseteq S \times Q$ and $R_2 \subseteq Q \times S$.

Claim 1: Let $(s, q) \in R_1$ and $\alpha \in \mathcal{I}^\star$, with $\delta(s, \alpha) = r$. Then $\mu(q, \alpha) = p$ and $(r, p) \in R_1$.

Proof of the Claim. An easy induction on $|\alpha| \geq 0$. △

CLAIM 2: Let $(s,q) \in R_1$ and $(q,s) \in R_2$. If $(s,p) \in R_1$, then $q = p$.

PROOF OF THE CLAIM. We show that p and q are not distinguishable. This implies that $p = q$, by the reducibility of N. If they were distinguishable, in N we would have $\mu(q,\alpha) = q'$, $\mu(p,\alpha) = p'$ and $q' \stackrel{x/a}{\rightarrow} q''$, $p' \stackrel{x/b}{\rightarrow} p''$, with $|\alpha| \geq 0$ and $a \neq b$.
Since $(s,q) \in R_2$ and $\mu(q,\alpha) = q'$, using Claim 1 we get $\delta(s,\alpha) = r$ and $(q',r) \in R_2$, for some $r \in S$. Then, $q' \stackrel{x/a}{\rightarrow} q''$ in N gives $r \stackrel{x/a}{\rightarrow} r'$ in M because R_2 is a simulation.
We also have $(s,p) \in R_1$, $\delta(s,\alpha) = r$ and $\mu(p,\alpha) = p'$. Claim 1 now gives $(r,p') \in R_1$. Since we already have $r \stackrel{x/a}{\rightarrow} r'$ in M, we conclude that $p' \stackrel{x/a}{\rightarrow} p'''$ in N, since R_1 is a simulation. But this contradicts $p' \stackrel{x/b}{\rightarrow} p''$ and $a \neq b$. △

CLAIM 3: $U(s_0) = U(q_0)$.

PROOF OF THE CLAIM. Let $\alpha \in U(s_0)$. An easy induction on $|\alpha| \geq 0$ gives $\alpha \in U(q_0)$. A symmetric argument shows that $U(q_0) \subseteq U(s_0)$. △

CLAIM 4: R_1 and R_2 are injective.

PROOF OF THE CLAIM. Let (s,q_1), $(s,q_2) \in R_1$. Since M is reduced, we get $\delta(s_0,\alpha) = s$, for some $\alpha \in \mathcal{I}^\star$. From Claim 3, $\mu(q_0,\alpha) = q_3$, for some $q_3 \in Q$. It is enough to show that $q_1 = q_3 = q_2$.
From Claim 1, we get $(s,q_3) \in R_1$ and $(q_3,s) \in R_2$. Since $(s,q_1) \in R_1$, Claim 2 gives $q_1 = q_2$. Similarly, from $(s,q_2) \in R_1$, Claim 2 now gives $q_2 = q_3$. △

CLAIM 5: R_1 and R_2 are total.

PROOF OF THE CLAIM. Let $s \in S$. Since M is reduced, we have $\delta(s_0,\alpha) = s$, for some $\alpha \in \mathcal{I}^\star$. Because $(s_0,q_0) \in R_1$, using Claim 1 we get $\mu(q_0,\alpha) = q$ and $(s,q) \in R_1$, showing that R_1 is total.
A similar argument shows that R_2 is also total. △

CLAIM 6: R_1 and R_2 are bijections.

PROOF OF THE CLAIM. By Claims 4 and 5, R_1 is total and injective. Then, $|S| \leq |Q|$. Similarly, $|Q| \leq |S|$ because R_2 is also total and injective. Thus, $|S| = |Q|$ and so R_1 and R_2 are also onto, and we may conclude that they are, in fact, bijections. △

CLAIM 7: R_1 and R_2 are isomorphisms.

PROOF OF THE CLAIM. By Claim 6 we already know that R_1 is a bijection, and that $(s_0,q_0) \in R_1$ because R_1 is a simulation. Let $s \stackrel{x/a}{\rightarrow} r$ in M. Since M is reduced, we get $\delta(s_0,\alpha) = s$ for some $\alpha \in \mathcal{I}^\star$. Then, Claim 1 gives $\mu(q_0,\alpha) = q$ and $(s,q) \in R_1$, for some $q \in Q$. Since R_1 is a simulation, we have $q \stackrel{x/a}{\rightarrow} p$ and $(r,p) \in R_1$, for some $p \in Q$. Since R_1 is in fact a function, we may write $q = R_1(s)$ and $p = R_1(r)$. This gives $R_1(s) \stackrel{x/a}{\rightarrow} R_1(r)$, and we may conclude that R_1 is an isomorphism.
A similar argument shows that R_2 is also an isomorphism. △

We can now state the desired result establishing that two FSMs are bi-similar if and only if they are isomorphic.

Theorem 3 *Let M and N be reduced FSMs. Then, M and N are bi-similar if and only if M and N are isomorphic.*

Proof If M and N are isomorphic then they are bi-similar by Lemma 2. The preceding series of claims establish the converse. □

The next corollary states the relationship between the isomorphism of FSMs and the notion of perfectness of test suites.

Corollary 4 *Let M be a reduced FSM and T be a test suite for M. Then T is perfect for M iff any reduced T-alike FSM is isomorphic to M.*

Proof From Theorem 1 and Theorem 3. □

The next example illustrates these results. We first note that in practice a test suite is usually generated using the specification as a guide, so that any generated test case does not block in the specification, that is, if T is the test suite generated and M is the specification, then we have $T \subseteq U(s_0)$. Under this condition, it is easy to check that for any implementation N, the condition $\big(U(s_0) \ominus U(q_0)\big) \cap T = \emptyset$ is equivalent to $T \subseteq U(q_0)$, that is, no test case blocks in the implementation either. In this example, therefore, we will assume $T \subseteq U(s_0)$. Now, given a specification M, a test suite T, and using Remark 2, we see that verifying if an implementation N is T-alike to M (see also Definition 6) is equivalent to checking whether $T \subseteq U(q_0)$, together with $\lambda(s, \sigma) = \tau(q, \sigma)$ for all $\sigma \in T$.

Then we see that, using Corollary 4, an algorithm to check for test suite perfectness might proceed by systematically visiting all implementations N such that $\lambda(s, \sigma) = \tau(q, \sigma)$ for all $\sigma \in T$. For each such implementation N we check if N is reduced. If that is not the case, N is discarded, otherwise we check if N and M are isomorphic. If an isomorphism is verified for all such reduced implementations N, we can claim that T is indeed perfect for M. If the isomorphism test fails in just one case, we can stop the process and claim the T is not perfect for M.

An algorithm for constructing all implementations N satisfying $\lambda(s, \sigma) = \tau(q, \sigma)$ for all $\sigma \in T$, was proposed in [BM14a] and extended in [BM14b, BM13]. That algorithm proceeds by constructing a so called T-tree, whose terminal leaves implicitly represent the desired implementations. We illustrate the main insights below, and for full details the mentioned references can be consulted.

Let M be the specification FSM depicted in Figure 2, and let $T = \{01000, 000, 10\}$ be a test suite for M. Clearly, M is reduced and $T \subseteq U(s_0)$. We want to verify if T is perfect for M. We obtain the T-tree partially depicted in Figure 3. The root is labeled by (s_0, q_0), at

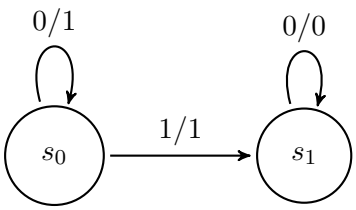

Figure 2: Specification FSM M.

level zero. It represents a trivial machine Z_0 with empty transition and output functions.

We then choose a new test case, say $\sigma = 01000$ in T. The first input symbol 0 of σ is used to obtain two descendant nodes at level one, namely Z_1 and Z_2. The partial machine Z_1 extends Z_0 by adding to $q_0 \overset{0/1}{\to} q_0$ to its transition set, whereas Z_2 extends Z_0 adding a new state q_1 and $q_0 \overset{0/1}{\to} q_1$ as a new transition to Z_0. The pair of states indicated next to each node in Figure 3 represents the states reachable from s_0 and q_0, respectively, when following the path from the root to the respective node.

The process continues at level 1 with the next input symbol in σ, namely the symbol 1, and attempting to extend all machines at that level. We obtain, for example, the partial machine Z_4 that extends Z_1 with a new state q_1 and a new transition $q_0 \overset{1/1}{\to} q_1$. When treating the last symbol of σ, namely 0, we would be at level 4, where we would have already the partial machine Z_5. Traversing the path from Z_0 down to Z_5 we can readily obtain all transitions in Z_5. Since the transition $q_1 \overset{0/0}{\to} q_1$ is already in Z_5, that machine would not be extended. At this point, we have exhausted all input symbols from σ. The cycle continues by taking another, yet unused, test case from T. Note that before starting on this new test case, we should relabel all state pairs at the current level to (s_0, q_0), since we must start at the initial states in both machines when considering a new test case. The whole tree-growing process terminates when all test cases have been treated. We finally note that some nodes are abandoned during the tree growing process. This is the case whenever we are at a node labeled (s, q), the current input symbol is x, and we already have transitions $s \overset{x/a}{\to} t$ in the specification and $q \overset{x/b}{\to} r$ in the partial machine at this node, with $a \neq b$. That node will be a dead, non-terminal, leaf of the tree. The symbol $\times$ in Figure 3 represents a branch where the tree-growing process is terminated because of such incompatibilities. Figure 3 only depicts part of the whole T-tree, and does not show the whole tree-growing process.

When the tree-growing process is terminated, we examine the terminal, non-dead, leaves in the tree. One such leaf is indicated as machine N in Figure 3, and we can read its transitions by traveling down to it from the root. We would obtain the machine depicted in Figure 4. We can readily see that it is a reduced machine and also that it is isomorphic to M.

In fact, it turns out that if we follow the tree-growing process to completion, all terminal non-dead leaves that represent reduced machines in the tree would be isomorphic to M. We could then claim that T is indeed a perfect test suite for M. Also notice that such an algorithm presented in [BM14a, BM14b] can be easily adapted in order to checking test suite perfectness, specifically when $T \not\subseteq U(s_0)$.

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 $n > k|S|$ is the number of states in implementation machines, 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.

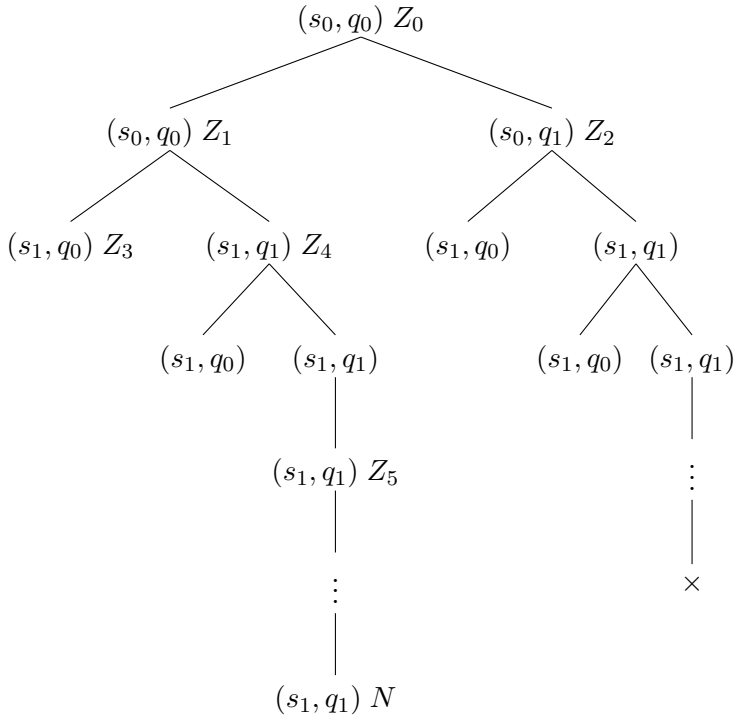

Figure 3: T-tree for FSM depicted in Figure 2.

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 sequences of symbols 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.

Remark 4 *Note that if M is a reduced FSM with at least two reachable states, 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.*

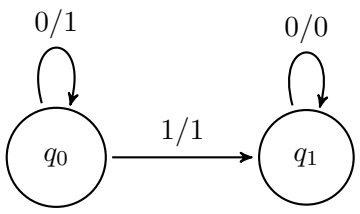

Figure 4: Implementation FSM N.

From this point on, we fix a reduced FSM M and a test suite T for M. Also, we fix $\sigma = x_0x_1 \cdots x_k$, $k \geq 0$, as a smallest non-extensible test case in T.

Remark 5 *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 using the same input and output alphabets, respectively $\mathcal{I}$ and $\mathcal{O}$, of M. A simple example illustrating the construction is presented right after Theorem 5. 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 $s \stackrel{y/b}{\rightarrow} r$ is not a distinguished transition of M, let $[s, i] \stackrel{y/b}{\rightarrow} [r, i]$ be a transition in N, for all i, $0 \leq i \leq k$.

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

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

(d) For the marked transition of M, $s_{k+1} \stackrel{z/a}{\rightarrow} s'$, we let $[s_{k+1}, k+1] \stackrel{z/b}{\rightarrow} [s', k+1]$, 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 $\pi : s \stackrel{\alpha/\omega}{\rightarrow} p$ in M and take $0 \leq i \leq k+1$. Then in N we must have $[s, i] \stackrel{\alpha/\omega'}{\rightarrow} [p, j]$ 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 $\pi : s \stackrel{\beta/\rho}{\rightarrow} r \stackrel{x/a}{\rightarrow} p$. The induction hypothesis gives $\pi_1 : [s, i] \stackrel{\beta/\rho'}{\rightarrow} [r, j]$ in N, with $j \geq i$.

If $j = k+1$, then items (c) and (d) in the construction of N give $[r, j] \stackrel{x/a'}{\rightarrow} [p, j]$ in N. Then, clearly, $[s, i] \stackrel{\alpha/\omega'}{\rightarrow} [p, j]$ 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 $r \stackrel{x/a}{\rightarrow} p$ 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 $[r,j] \stackrel{x/a'}{\to} [p,\ell]$ in N where $\ell = j$ or $\ell = j+1$. Hence, $[s,i] \stackrel{\alpha/\omega'}{\to} [p,j]$ 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, from 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 $\pi : [s,i] \stackrel{\alpha/\omega}{\to} [p,j]$ in N. Then we have: (i) $j \geq i$, (ii) $\sigma_{i,j}$ is embedded in α, and (iii) $s \stackrel{\alpha/\omega'}{\to} p$ 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. Following 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 $\sigma_{i,\ell}$ is 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 will 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 $s_0 \stackrel{\alpha/\omega}{\rightarrow} s \stackrel{x/a}{\rightarrow} r$ in M and $[s_0, 0] \stackrel{\alpha/\omega}{\rightarrow} [q, i] \stackrel{x/b}{\rightarrow} [p, j]$ in N, with $a \neq b$. Fact 2 gives $s_0 \stackrel{\alpha/}{\rightarrow} q$ in M. But we already have $s_0 \stackrel{\alpha/}{\rightarrow} s$ in M, and so we conclude that $s = q$. Using Fact 2 again, from $s \stackrel{x/}{\rightarrow} r$ in M and $[s, i] \stackrel{x/}{\rightarrow} [p, j]$ in N we get $p = r$. We can now write $\pi : [s, i] \stackrel{x/b}{\rightarrow} [r, j]$ 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 5 *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 4 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 2 again. 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 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 [BM14b, BM13].

Now take $\sigma = 100$ as the shortest test case in T that is non-extensible in T. We apply items (a) to (d) of the construction of N, thus obtaining a machine 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 machine N with the marked transition $[s_3, 3] \stackrel{0/1}{\to} [s_3, 3]$ as required by item (d). Machine N is depicted in Figure 5. 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 them in order to obtain a reduced FSM as depicted in Figure 6. Note that we have renamed states by letting $q_0 = [s_0, 0]$, $q_1 = [s_1, 1]$,

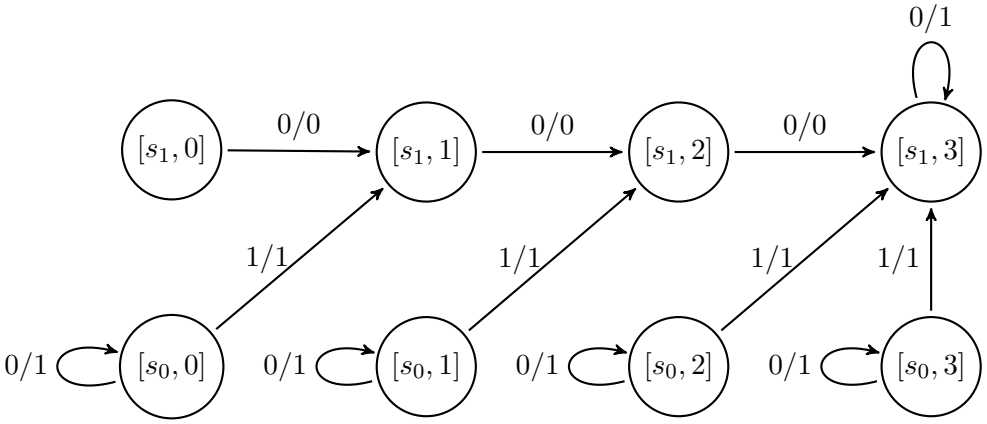

Figure 5: A candidate implementation N.

$q_2 = [s_1, 2]$, and $q_3 = [s_1, 3]$.

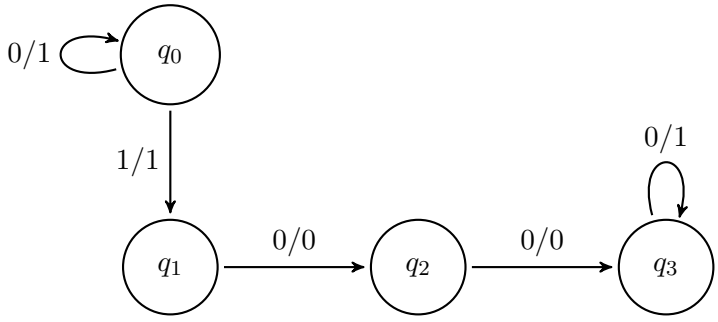

Figure 6: A reduced candidate implementation N.

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 $U(s_0) \subseteq U(q_0)$. We conclude that T is not 4-complete for M, and so it is also not 8-complete for M.

5 Conclusions

This work has shown some important properties related to the completeness of test suites. We have used bi-simulations to characterize test suite perfectness, a new notion of completeness when blocking test cases can be present in more general scenarios. We then proved that a test suite is perfect for a specification when implementations are bi-similar to the specification. Moreover, when the implementation machine has as many states as in the specification, if a test suite is perfect then the implementation is unique and isomorphic to the specification.

We also established an upper bound in the maximum size of implementations that can be put under test. The bound is based on the test suite size and the number of states in the given specification. In this scenario, no test suite can be complete when considering the classical notion of test suite completeness.

For future work one might consider developing a tool for automatically checking test suite perfectness.

References

[BM13] Adilson Luiz Bonifacio and Arnaldo Vieira Moura. Partial fsm models and completeness with blocking test cases. Technical Report IC-13-33, Institute of Computing, University of Campinas, November 2013.

[BM14a] Adilson Luiz Bonifacio and Arnaldo Vieira Moura. On the Completeness of Test Suites. In *Proceedings of the 29th ACM Symposium on Applied Computing (ACM SAC)*, volume 2, pages 1287–1293. ACM, march 2014.

[BM14b] Adilson Luiz Bonifacio and Arnaldo Vieira Moura. Test suite completeness and partial models. In D. Giannakopoulou and G. SalaÃ$\frac{1}{4}$n, editors, *Proceedings of the 12th International Conference on Software Engineering and Formal Methods (SEFM)*, volume 8702 of *Lecture Notes in Computer Science*, pages 96–110, Grenoble, France, 01–05, sep 2014. Springer Verlag.

[BMdSS12] 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.

[DEFY05] Rita Dorofeeva, Khaled El-Fakih, and Nina Yevtushenko. An improved conformance testing method. In *FORTE*, pages 204–218, 2005.

[dSSPY09] Adenilso da Silva Simao, Alexandre Petrenko, and Nina Yevtushenko. Generating reduced tests for fsms with extra states. In *TestCom/FATES*, pages 129–145, 2009.

[Gil62] A. Gill. *Introduction to the theory of finite-state machines*. McGraw-Hill, New York, 1962.

[HU02] Robert M. Hierons and Hasan Ural. Reduced length checking sequences. *IEEE Trans. Comput.*, 51(9):1111–1117, September 2002.

[PB96] A. Petrenko and G. V. Bochmann. On fault coverage of tests for finite state specifications. *Computer Networks and ISDN Systems*, 29:81–106, 1996.

[PY00] Alex Petrenko and Nina Yevtushenko. On test derivation from partial specifications. In *In FORTE*, pages 85–102, 2000.

[SP10] Adenilso da Silva Simao and Petrenko Petrenko. Checking completeness of tests for finite state machines. *IEEE Trans. Computers*, 59(8):1023–1032, 2010.

[SPY12] Adenilso Simao, Alexandre Petrenko, and Nina Yevtushenko. On reducing test length for fsms with extra states. *Softw. Test. Verif. Reliab.*, 22(6):435–454, September 2012.

[UWZ97] Hasan Ural, Xiaolin Wu, and Fan Zhang. On minimizing the lengths of checking sequences. *IEEE Trans. Comput.*, 46(1):93–99, January 1997.

[YPvB94] Ming Yu Yao, Alexandre Petrenko, and Gregor von Bochmann. Fault coverage analysis in respect to an fsm specification. In *INFOCOM*, pages 768–775, 1994.