INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Exponentially more Succinct Test Suites

A. Bonifácio   *A. Moura*   *A. Simão*

Technical Report - IC-09-07 - Relatório Técnico

March - 2009 - Março

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

Exponentially more Succinct Test Suites

Adilson Luiz Bonifácio[*] Arnaldo Vieira Moura[†] Adenilso da Silva Simão[‡]

Abstract

We present a generalized test case generation method, or G-method, extending some previous work [1]. Although inspired on the W-method [3], the G-method, in contrast, allows for test case suite generation even in the absence of characterization sets for the specification models. Instead, the G-method relies on knowledge about the index of certain equivalences induced in the implementation models. We show that the W-method can be derived from the G-method as a particular case. Moreover, we discuss some naturally occurring infinite classes of FSM models over which the G-method generates test suites that are exponentially more compact than those produced by the W-method. Proofs for important results are presented in detail.

1 Introduction

There are many approaches described in the literature for the automatic generation of test case suites that can be used to verify the correctness of reactive and critical system implementations. It is desirable that such techniques and methods be efficient and accurate in terms of fault coverage. Many among such approaches use formal models as a foundation, giving rise to model-based testing strategies that allow for the automatic generation of efficient test case suites from mathematical models of system requirements, and formally specified system functionalities.

In practice, the aim of system testing is not to demonstrate an equivalence between a specification and a number of implementations [7, 20, 22]. Indeed, in the majority of real application cases it is infeasible to establish equivalence using test generation methods. As an alternative, one can substitute equivalence for a notion of conformance testing, a more relaxed concept than full equivalence. Here, the aim is to demonstrate how to test whether certain implementation behaviors conform to desired behaviors extracted from specifications [22].

A number of model-based test generation methods for conformance testing of critical and reactive systems have been proposed [4, 15, 23]. Many such methods use Finite State

[*]Instituto de Computação, Unicamp, P.O. 6176 – Campinas – Brazil – 13081-970, Suporte financeiro do CNPq, processo 141978/2008-2

[†]Instituto de Computação, Unicamp, P.O. 6176 – Campinas – Brazil – 13081-970, Suporte financeiro do CNPq, processo 472504/2007-0

[‡]Instituto de Ciências Matemáticas e de Computação - ICMC-USP, Universidade de São Paulo, P.O. 668 – São Carlos – Brazil – 13560-970

Machines (FSMs) [2, 5, 9, 10, 11, 13, 19, 24] as their basic foundations. One of the most well-known of these test generation methods is the W-method [3], which uses the notion of characterization sets. The W-method assumes deterministic, minimal and complete FSMs, and many variations have been developed around its main ideas [11, 12, 14, 18].

In this article, we generalize the W-method, deriving test suites which can be made m-complete. A test suite is m-complete if it guarantees a complete fault coverage [17], while considering deterministic FSM implementations with up to m states. The new algorithm, here named the G-method, can generate test case suites even in the absence of characterization sets [1]. Further, the G-method can be specialized in a way so as to conform to the original W-method, thus demonstrating that the latter is a particular case of the G-method.

The G-method constructs test case suites based on characteristics of the specification and on information, supplied by the user, about the set of implementations targeted for testing, information such as a lower bound on the number of certain equivalence classes induced in the implementation models. This knowledge can be obtained from various sources, *e.g.*, from regression test suites, design standard enforcement, or from the tester's expertise. Provided that such information is available, compact test suites can be generated. Although stringent values for such parameters will drive the G-method toward m-completeness, while lowering its efficiency, other judicious choices might give rise to very compact test suites.

In fact, in this paper we show that the G-method can outperform the W-method in an infinite class of naturally occurring FSM specification and implementation models. More specifically, we show that the G-method generates test case suites that are exponentially more succinct than those obtained with the original W-method, when applied over such infinite families.

Given that the G-method can generate more efficient test suites, an alternative testing strategy can be devised. First, we can generate compact test suites using the G-method, with appropriate parameters. Then, we can apply such test suites to the set of implementation candidates, assuming a fixed upper bound on the number of states for the implementation models. If some target implementations pass this first testing stage, one could produce more stringent test suites, and use them to test the remaining implementation models in a second step. In this second step one could, with extra effort, construct complete test suites using the same G-method with stronger parameters, or even using the original W-method, or any other complete method.

The paper is organized as follows. We review some basic concepts in Section 2. In Section 3 we introduce equivalence in FSMs and stratified families of sets. The generation of complete test suites is presented in Section 4. In Section 5 we reconsider characterization sets, in the sequel, show how to obtain the original W-method in Section 6. In Section 7 we present the algorithm for the generalized method, and illustrate its usefulness by mean of an example. In Section 8 we discuss infinite families of FSM models over which the G-method outperforms the W-method. In Section 9 we describe related works. Finally, in Section 10, we state some concluding remarks.

2 Basic Concepts

This section reviews the FSM model, and some important related notions. Further we also present the cover property essentially for test case generation based on FSMs.

2.1 Finite State Machines

The basic model used to capture a system behavior is the FSM. Formally, a FSM [8] is a system $M = (X, Y, S, s_0, \delta, \lambda)$ given by:

- a finite input alphabet, X;
- a finite output alphabet, Y;
- a finite set of states, S;
- an initial state $s_0 \in S$; and
- output and transition functions, respectively, $\lambda : X \times S \to Y$ and $\delta : X \times S \to S$.

Note that such a machine is complete, i.e for each state s of M, there is a transition from s with input symbol a, for every $a \in X$, and deterministic, i.e. a FSM does not allow two different transitions going out of the same state with identical input symbols.

Successive applications of the transition function δ give rise to the extended transition function $\widehat{\delta} : X^\star \times S \to S$, defined by

$$\begin{array}{rcl} \widehat{\delta}(\epsilon, s) & = & s, \\ \widehat{\delta}(a\rho, s) & = & \widehat{\delta}(\rho, \delta(a, s)), \text{where } a \in X \text{ and } \rho \in X^\star. \end{array}$$

Here, ϵ will denote the empty word. For convenience, if $\widehat{\delta}(\rho, s_1) = s_2$ we also write $s_1 \xrightarrow{\rho} s_2$.

We extend λ to $\widehat{\lambda} : X^\star \times S \to Y^\star$ thus

$$\begin{array}{rcl} \widehat{\lambda}(\epsilon, s) & = & \epsilon, \\ \widehat{\lambda}(a\rho, s) & = & \lambda(a, s)\widehat{\lambda}(\rho, \delta(a, s)), \text{with } a \in X,\ \rho \in X^\star. \end{array}$$

Henceforth, unless mention to the contrary, we will assume that M and M' denote FSMs in the form $M = (X, Y, S, s_0, \delta, \lambda)$ and $M' = (X, Y', S', s'_0, \delta', \lambda')$. Note that M and M' have the same input alphabet.

The reachability notion expresses the idea of starting at the initial state, traversing some transitions, and reaching a target state.

Definition 1 *A state s in a FSM M is reachable if and only if there exists $\rho \in X^\star$ such that $\widehat{\delta}(\rho, s_0) = s$.* ▮

We also say that $\widehat{\lambda}(\rho, s)$ is the behavior of M from state s over the input sequence ρ. The behavior of M over ρ is simply the behavior of M from s_0 over ρ. A sequence ρ distinguishes two states s_1 and s_2 of M if ρ gives distinct behaviors for s_1 and s_2, that is, if $\widehat{\lambda}(\rho, s_1) \neq \widehat{\lambda}(\rho, s_2)$.

2.2 Cover sets

The notion of transition cover is an important concept needed in several methods for generating test sequences. A transition cover is given by a cover set over a FSM. Hence, let M be a FSM. A cover set $P \subseteq X^\star$ is required to exercise every transition in M, *i.e.*, for every transition $\delta(a, s) = r$ in M there must be $\rho, \rho a \in P$ such that $\widehat{\delta}(\rho, s_0) = s$. In this way, we can obtain a behavior of M that reaches state s, and terminates by traversing the specific edge from s to r, labeled by a.

The cover set notion is formalized next.

Definition 2 *A set of input sequences $C \subseteq X^\star$ is a cover set for a FSM M if for every pair of states $s, r \in S$ and every input symbol $a \in X$, with $\delta(a, s) = r$, there exist $\rho, \rho a \in C$ such that $\widehat{\delta}(\rho, s_0) = s$.* ▮

A cover set can be obtained by constructing a labeled tree for M. A labeled tree is a system $T = (N, A, l_v, l_e)$, where N is a set of nodes, A is the set of edges, and $l_v : N \to S$ and $l_e : A \to X$ are labeling functions of nodes and edges, respectively. The nodes in the tree will be labeled by states of M and edges will be labeled by symbols from X.

Construction 3 *A labeled tree for M, $T = (N, A, l_v, l_e)$, can be constructed as follows:*

1. *Initiate with $N = \{n_0\}$, $A = \emptyset$, $l_v(n_0) = s_0$ and $l_e = \emptyset$, where s_0 is the initial state of M and n_0 is the root of T. We say that n_0 has level zero in T.*

2. *Inductively, suppose T is already constructed up to level $k \geq 0$. Level $k + 1$ is constructed by inspecting of nodes in level k from left to right:*

   (a) *let $n \in N$ be the next node to be inspected.*

   (b) *if there already exists $m \in N$ with $l_v(m) = l_v(n)$, and m is at some level $l < k$ in T, then node n is ignored, and we take the next node at level k. Otherwise, for every input $a \in X$ and every $r \in S$ with $r = \delta(a, l_v(n))$, we add a new node n' to N, a new edge (n, n') to A, and define $l_v(n') = r$ and $l_e(n, n') = a$. We then proceed to the next node in level k.*

3. *Step 2 is repeated if new nodes were added to T in the last iteration; otherwise, T is completed.* ▮

The process will always terminate since the set of states in M is finite. Depending on how the symbols from X are selected, different trees can be obtained (see step (2b) in Construction 3).

The next lemma expresses a simple fact.

Lemma 4 *Let $T = (N, A, l_v, l_e)$ be a labeled tree for a FSM M. Let $n \in N$ and let α be the sequence of edge labels in the simple path from the root to n in T. Then, $\widehat{\delta}(\alpha, s_0) = l_v(n)$.*

Proof By a simple induction on the level of n in T. ▮

The next definition shows how to construct a required cover set.

Definition 5 *Let T be a labeled tree for M. The set P_T is defined by all words $\alpha \in X^\star$ which label paths in T, starting at the root.* ▮

Note that $\epsilon \in P_T$. When T is clear from the context, we will use the simplified notation P instead of P_T.

We can now show that P_T, from Definition 5, is a cover set for machine M. Before that, we need a property of labelled trees.

Lemma 6 *Let $T = (N, A, l_v, l_e)$ be a labeled tree for a FSM M, as given by Construction 3. Let P_T be the set obtained as in Definition 5. Let $\rho \in X^\star$ and $s \in S$ be such that $\widehat{\delta}(\rho, s_0) = s$. Then, there exists a node $n \in N$ with $l_v(n) = s$. Furthermore, there exists a sequence $\alpha \in P_T$ with $\widehat{\delta}(\alpha, s_0) = s$ and such that for every edge $\delta(a, s) = r$ we have $\alpha a \in P_T$.*

Proof By induction on $|\rho| = 0$. If $|\rho| = 0$, then $\rho = \epsilon$ and $s = s_0$. Let n be the root of T. Then $l_v(n) = s_0 = s$, by step (1) of Construction 3. Now, choosing $\alpha = \epsilon$, we have $\alpha \in P_T$, with $\widehat{\delta}(\alpha, s_0) = s_0 = s$. Moreover, if $\delta(a, s) = r$ is an edge of M, then step (2b) of Construction 3 will add a node m to N, at level 1, and an edge (n, m) to A, labeling them $l_v(m) = r$ and $l_e(n, m) = a$. This comes from the fact that n is at the level zero in T (step (1) of Construction 3) and, obviously, there is no node at an inferior level. Clearly, the sequence of edge labels in the path from the root n up to m in T is a. Then, $a \in P_T$. Hence, $\alpha a \in P_T$, because $\alpha a = a$.

Assume such result holds for every ρ with $|\rho| \leq k$, where $k \geq 0$. Let $\rho = \sigma b$, with $\sigma \in X^k$, $b \in X$ and $\widehat{\delta}(\rho, s_0) = s$. We have some $s_1 \in S$ such that $\widehat{\delta}(\sigma, s_0) = s_1$ and $\delta(b, s_1) = s$. By induction, there is a node $n \in N$ with $l_v(n) = s_1$. Let $h \geq 0$ be the level of n in T. Without loss of generality, we can assume that h is minimal. In this way, when n is examined during the step 2 in Construction 3, by using the minimality of h, given the edge $\delta(b, s_1) = s$ and knowing that $l_v(n) = s_1$, a new node n' will be added to N, at level $h+1$, with $l_v(n') = s$. Therefore, n' satisfies the first assertive of the lemma. Now we know there is a node in N whose label is s. Again, let $h' \geq 0$ be the lowest level of a node $m \in N$ such that $l_v(m) = s$. Let α be a sequence of edge labels in the path from the root up to m in T. From Lemma 4 we have $\widehat{\delta}(\alpha, s_0) = l_v(m) = s$ and, from the construction of P_T we have $\alpha \in P_T$. Consider an edge $\delta(a, s) = r$. The minimality of h' guarantees that step (2b) of Construction 3 will add a new node m' to N, at level $h' + 1$, and an edge (m, m') to A, with $l_e(m, m') = a$. By construction of P_T, $\alpha a \in P_T$, thus proving the second assertive of the lemma, and completing the proof. ▮

Now we can enunciate the cover set property.

Corollary 7 *Let $T = (N, A, l_v, l_e)$ be the labeled tree for a FSM M, as given by Construction 3. Let P_T be the set obtained as in Definition 5. If every state of M is reachable, then the set $P_T \subseteq X^\star$ is a cover set for M.*

Proof Let $\delta(a, r) = s$ be an edge. As r is reachable, we have $\widehat{\delta}(\rho, s_0) = r$, for some $\rho \in X^\star$. By Lemma 6, we have α, $\alpha a \in P_T$ with $\widehat{\delta}(\alpha, s_0) = r$, for some $\alpha \in X^\star$. Thus, P_T is a cover set for M. ▮

3 Equivalences and Stratified Families

This section deals with the concept of equivalence over machines, and also presents the concept of stratified families used to obtain state partitions.

3.1 Equivalence relation over FSMs

The concept of equivalence starts with the notion of state equivalence relations induced by the transition functions of FSMs. The next definition exposes those notions in a general context.

Definition 8 *Let M and M' be two FSMs over the same input alphabet, X, and let s and s' be states of M and M', respectively.*

1. *Let $\rho \in X^\star$. We say that s is ρ-equivalent to s' if $\widehat{\lambda}(\rho, s) = \widehat{\lambda}'(\rho, s')$. In this case, we write $s \approx_\rho s'$. Otherwise, s and s' are ρ-distinguishable and we write $s \not\approx_\rho s'$.*
2. *Let $K \subseteq X^\star$. We say that s is K-equivalent to s' if s is ρ-equivalent to s', for every $\rho \in K$. In this case, we write $s \approx_K s'$. Otherwise, s and s' are K-distinguishable and we write $s \not\approx_K s'$.*
3. *Let $k \geq 0$. We say that s is k-equivalent to s' if s is X^k-equivalent to s'. Otherwise, s and s' are k-distinguishable. We write, respectively, $s \approx_k s'$ and $s \not\approx_k s'$.*
4. *State s is equivalent to s' if s is k-equivalent to s', for every $k \geq 0$. Otherwise, s and s' are distinguishable. We write, respectively, $s \approx s'$ and $s \not\approx s'$.* ∎

We will avoid overloading the notation by indicating M and M' explicitly, *e.g.*, in the form $\approx_k^{M,M'}$, since both machines will always be clear from the context. Definition 8 also applies when M and M' are the same machine. In this case, it is easy to verify that all relations defined above are, in fact, equivalence relations over the state set of the machine. Hence, each such equivalence relation $\approx_Z$ gives rise to a partition $[Z]$ of the state set S.

Definition 9 *Let M be a FSM. The index of M, ι_M, is the number of equivalence classes induced by the $\approx$ relation over the states of M.* ∎

Clearly, we will always have $1 \leq \iota_M \leq |S|$, where S is the state set of M.

The next lemma gathers some simple observations.

Lemma 10 *Let M and M' be two FSMs with states s and s', respectively.*

1. *Let $K \subseteq X^\star$. If $s \approx_K s'$, then $s \approx_L s'$, for every L with $L \subseteq K$. On the other hand, if $s \not\approx_K s'$, then $s \not\approx_L s'$, for every L with $K \subseteq L$.*
2. *Let $k \geq 0$. If $s \approx_k s'$ then $s \approx_l s'$ for every l with $l \leq k$. On the other hand, if $s \not\approx_k s'$, then $s \not\approx_l s'$, for every l with $l \geq k$.*
3. *Let $K, L \subseteq X^\star$. If $s \not\approx_K s'$, then $s \not\approx_{KL} s'$, for every $L \neq \emptyset$.*

Proof Trivial. ∎

3.2 Equivalence induced by Stratified Families

Now we present the notion of stratified families, sets of input sequences to induce state partitions in FSM models. In the sequel, we will be considering such specific sets of input sequences.

Definition 11 *Let $Z_i \subseteq X^\star$, $i \geq 0$, where X is an alphabet. We say that $\{Z_i\}_{i\geq 0}$ is a stratified family over X if*

1. *$Z_0 \neq \emptyset$; and*
2. *$(X \cup \{\epsilon\})Z_i = Z_{i+1}$, for every $i \geq 0$.* ▮

It is easy to see that these properties are independent of each other.

Another characterization for stratification is given as follows.

Proposition 12 *Let $Z_i \subseteq X^\star$, $i \geq 0$, where X is an alphabet and with $Z_0 \neq \emptyset$. Then, the family $\{Z_i\}_{i\geq 0}$ is stratified if and only if $Z_k = \bigcup_{j=0}^{k} X^j Z_0$ for every $k \geq 0$.*

Proof Assume that $Z_k = \bigcup_{j=0}^{k} X^j Z_0$, for every $k \geq 0$. Then,

$$\begin{aligned}(X \cup \{\epsilon\})Z_k &= XZ_k \bigcup Z_k \\ &= X(\bigcup_{0\leq j\leq k} X^j Z_0) \bigcup (\bigcup_{0\leq j\leq k} X^j Z_0) \\ &= (\bigcup_{1\leq j\leq k+1} X^j Z_0) \bigcup (\bigcup_{0\leq j\leq k} X^j Z_0) \\ &= \bigcup_{0\leq j\leq k+1} X^j Z_0 = Z_{k+1}.\end{aligned}$$

Since we have $Z_0 \neq \emptyset$, the stratification is established.

Now assume that $\{Z_i\}_{i\geq 0}$ is a stratified family. When $k = 0$, it is immediate that $Z_k = \bigcup_{j=0}^{k} X^j Z_0$. Continuing by induction, we assume that the result holds for $k-1$, where $k \geq 1$, and show that $Z_k = \bigcup_{j=0}^{k} X^j Z_0$.

Let $z \in Z_k$. Then, from Definition 11, $z \in (X \cup \{\epsilon\})Z_{k-1}$. If $z \in Z_{k-1}$ then the induction hypothesis guarantees that $z \in \bigcup_{j=0}^{k-1} X^j Z_0$, and then $z \in \bigcup_{j=0}^{k} X^j Z_0$. On the other hand, if $z \in XZ_{k-1}$, then $z = aw$, with $a \in X$ and $w \in Z_{k-1}$. From the induction hypothesis, $w \in \bigcup_{j=0}^{k-1} X^j Z_0$, and then $z \in \bigcup_{j=1}^{k} X^j Z_0$, and so, $z \in \bigcup_{j=0}^{k} X^j Z_0$. Therefore, $Z_k \subseteq \bigcup_{j=0}^{k} X^j Z_0$. Now let $z \in \bigcup_{j=0}^{k} X^j Z_0$. If $z \in \bigcup_{j=0}^{k-1} X^j Z_0$, the induction hypothesis gives $z \in Z_{k-1}$. From Definition 11(2), we obtain $z \in Z_k$. If $z \in X^k Z_0$, then $z = aw$ with $a \in X$ and $w \in X^{k-1} Z_0$. From the induction hypothesis, $w \in Z_{k-1}$ and, consequently, $z \in XZ_{k-1}$. Again, from Definition 11(2) we obtain $z \in Z_k$. Therefore, $\cup_{j=0}^{k} X^j Z_0 \subseteq Z_k$. We conclude that $Z_k = \cup_{j=0}^{k} X^j Z_0$, extending the induction. ▮

The next result guarantees that certain sequences always have continuations in some of the Z_k sets.

Lemma 13 *Let $\{Z_i\}_{i\geq 0}$ be a stratified family over X and let $k \geq 0$. Then*

1. *$Z_k \subseteq Z_j$, for every $j \geq k$; and*
2. *For every $\alpha \in X^j$, with $0 \leq j \leq k$, there exists $\beta \in X^\star$ such that $\alpha\beta \in Z_k$.*

Proof From Proposition 12, we deduce that $Z_i \subseteq Z_{i+1}$, for every $i \geq 0$. A simple induction establishes item (1). For item (2), since $Z_0 \neq \emptyset$, we take $\gamma \in Z_0$. Since $j \leq k$, we take $\sigma \in X^{k-j}$. Hence, $\alpha\sigma\gamma \in X^k Z_0$. From Proposition 12 we conclude $\alpha\sigma\gamma \in Z^k$. ∎

Let M be a FSM and let $Z \subseteq X^\star$ be a set of input sequences. We indicate by $[Z]$ the partition induced by Z (see observation after Definition 8) over the states of M, i.e, $s \approx_Z r$ if and only if $s, r \in \omega$, for some $\omega \in [Z]$. Let $[Z_1]$ and $[Z_2]$ be two partitions over S. Then we say that $[Z_2]$ *refines* $[Z_1]$ if and only if for all $\omega_2 \in [Z_2]$ there exists some $\omega_1 \in [Z_1]$ such that $\omega_2 \subseteq \omega_1$.

The next result expresses properties of these partitions.

Lemma 14 *Let $\{Z_i\}_{i\geq 0}$ be a stratified family over the alphabet X of a FSM M. Then*

1. *$[Z_{i+1}]$ refines $[Z_i]$, for every $i \geq 0$; and*
2. *if $|[Z_k]| = |[Z_{k+1}]|$ for some $k \geq 0$, then we must have $[Z_k] = [Z_{k+1}] = [Z_{k+2}]$.*

Proof We show each item, in turn.

For item (1), assume that it does not hold for some $i \geq 0$. Then we will have states s and r such that $s \approx_{Z_{i+1}} r$ and $s \not\approx_{Z_i} r$. From Lemma 10(1) and Lemma 13(1) we deduce $s \not\approx_{Z_{i+1}} r$, a contradiction.

Now we verify item (2). From item (1), we know that $[Z_{k+1}]$ refines $[Z_k]$. Then $[Z_k] = [Z_{k+1}]$, otherwise we would have $|[Z_k]| < |[Z_{k+1}]|$. Again continuing by contradiction, assume that $[Z_{k+1}] \neq [Z_{k+2}]$. Since $[Z_{k+2}]$ refines $[Z_{k+1}]$, we will have states r and s such that $s \not\approx_{Z_{k+2}} r$ and $s \approx_{Z_{k+1}} r$. Hence, we obtain $\rho \in Z_{k+2}$, with $\rho = a\beta$ and $a \in X$, and such that $s \not\approx_{a\beta} r$. We also conclude that $a\beta \notin Z_{k+1}$, otherwise we would have the contradiction $s \not\approx_{Z_{k+1}} r$. Therefore, from Definition 11(2), we deduce $a\beta \in XZ_{k+1}$, and so, $\beta \in Z_{k+1}$.

Let $s_1, r_1 \in S$ with $s_1 = \delta(a, s)$, $r_1 = \delta(a, r)$. If $s_1 \not\approx_{Z_{k+1}} r_1$ then $s_1 \not\approx_{Z_k} r_1$, because we already know that $[Z_k] = [Z_{k+1}]$. Hence, we would have $\gamma \in Z_k$ with $\widehat{\lambda}(\gamma, s_1) \neq \widehat{\lambda}(\gamma, r_1)$. From Definition 11(1) we have $XZ_k \subseteq Z_{k+1}$, and then $a\gamma \in Z_{k+1}$. But,

$$\begin{aligned}
\widehat{\lambda}(a\gamma, s) &= \lambda(a, s)\widehat{\lambda}(\gamma, s_1)\\
\widehat{\lambda}(a\gamma, r) &= \lambda(a, r)\widehat{\lambda}(\gamma, r_1).
\end{aligned}$$

Then we have $\widehat{\lambda}(a\gamma, s) \neq \widehat{\lambda}(a\gamma, r)$, thus forcing the contradiction $s \not\approx_{Z_{k+1}} r$. We conclude that $s_1 \approx_{Z_{k+1}} r_1$.

Since $\beta \in Z_{k+1}$, we deduce $s_1 \approx_\beta r_1$. Again,

$$\begin{aligned}
\widehat{\lambda}(a\beta, s) &= \lambda(a, s)\widehat{\lambda}(\beta, s_1)\\
\widehat{\lambda}(a\beta, r) &= \lambda(a, r)\widehat{\lambda}(\beta, r_1),
\end{aligned}$$

and, since we already have $\widehat{\lambda}(\rho, s) \neq \widehat{\lambda}(\rho, r)$, we conclude that $\lambda(a, s) \neq \lambda(a, r)$. From $a \in X$ and Lemma 13(2) we infer $\sigma \in X^\star$ with $a\sigma \in Z_{k+1}$. Hence, we have $s \not\approx_{a\sigma} r$, contradicting $s \approx_{Z_{k+1}} r$. ▮

The next result gives the equality of successive partitions.

Corollary 15 *Let $\{Z_i\}_{i\geq 0}$ be a stratified family over the input alphabet X of a FSM M. If $|[Z_k]| = |[Z_{k+1}]|$ for some $k \geq 0$, then $[Z_k] = [Z_{k+l}]$ for every $l \geq 0$.*

Proof When $l = 0$, the result is immediate. When $l = 1$ or $l = 2$, the result follows directly from Lemma 14(2). Assume the result holds for every j, $0 \leq j \leq l$, with $l \geq 2$. We want to show that the result holds for $l + 1$. From the induction, we have $[Z_k] = [Z_{k+l}]$ and $[Z_k] = [Z_{k+l-1}]$. Hence, $[Z_{k+l-1}] = [Z_{k+l}]$. Using Lemma 14(2), we obtain $[Z_{k+l-1}] = [Z_{k+l}] = [Z_{k+l+1}]$. Hence, $[Z_k] = [Z_{k+l+1}]$, as required. ▮

Now let M be a FSM with m states. Suppose we have a stratified family for X, $\{Z_i\}_{i\geq 0}$, in which Z_0 partitions the states of M in $n \leq m$ equivalence classes. The next lemma establishes the basic result about partitions over states of M induced by the Z_i sets, for $i \geq 0$.

Lemma 16 *Let M be a FSM with index m. Let $\{Z_i\}_{i\geq 0}$ be a stratified family for X such that Z_0 partitions the states of M in at least $n \leq m$ equivalence classes. Then $|[Z_i]| \geq n + i$, for every i, with $0 \leq i \leq m - n$.*

Proof When $i = 0$ we have $n + i = n$ and, from the hypothesis, $|[Z_0]| \geq n$, establishing the base. Assume the result for every j, $0 \leq j \leq i$, with $i < m - n$. We show that the result holds for $i + 1$. If $|[Z_i]| \geq n + i + 1$ then $|[Z_{i+1}]| \geq n + i + 1$ (from Lemma 14(1)), and the induction is extended in this case.

Now, let $|[Z_i]| < n + i + 1$. From the induction hypothesis we conclude that $|[Z_i]| = n + i$. Since $m \geq n + i + 1$ is the index of M, there exist nonequivalent states in M, r and s, with $r \approx_{Z_i} s$. Then, $s \not\approx_{X^k} r$, for some $k \geq 0$ (see Definition 8). From Lemma 13(2), we conclude $s \not\approx_{Z_k} r$. If $k \leq i$, Lemma 13(1) would force $Z_k \subseteq Z_i$. Using Lemma 10(1) we would have $s \not\approx_{Z_i} r$, a contradiction. Hence, $k > i$.

If $|[Z_i]| = |[Z_{i+1}]|$ then, by Corollary 15, we get $Z_i = Z_k$, forcing again the contradiction $s \not\approx_{Z_i} r$. Since $[Z_{i+1}]$ refines $[Z_i]$, we can not have $|[Z_{i+1}]| < |[Z_i]|$. We conclude that $|[Z_{i+1}]| > |[Z_i]|$. But, since $|[Z_i]| = n + i$, we deduce the result desired, that is, $|[Z_{i+1}]| \geq n + i + 1$. ▮

Using this result, it will be easy to confirm that some $Z \in \{Z_i\}_{i\geq 0}$ will distinguish every pair of nonequivalent states.

Corollary 17 *Let M be a FSM with index m. Let $\{Z_i\}_{i\geq 0}$ be a stratified family for X such that Z_0 partitions the states of M in at least $n \leq m$ equivalence classes. Then Z_{m-n} will distinguish every pair of nonequivalent states of M.*

Proof From Lemma 16, it follows that $|[Z_{m-n}]| \geq n + (m - n) = m$. Since $[Z_{m-n}]$ is the partition induced by Z_{m-n}, we conclude that Z_{m-n} partitions states of M in m classes. Since M has index m, we conclude that Z_{m-n} will distinguish every pair of nonequivalent states of M. ▮

4 Generating a m-complete Test Suite

Based on previous sections we can now define the notion of m-complete test suites. Let M and M' be two FSMs operating over the same alphabet X. Machine M represents a specification and M' represents a possible implementation of M. We want to obtain a set $K \subseteq X^\star$ such that $s_0 \not\approx s'_0$ if and only if $s_0 \not\approx_K s'_0$. Such a set K is a m-complete test suite, where m is an upper bound on the index of M'. Given K, if we want to test whether M and M' have distinct behaviors, it is enough to apply the sequences in K to both machines and compare the corresponding output sequences.

We obtain the required set by combining a cover set for M with a stratified family for M'. The next lemma establishes an auxiliary result.

Lemma 18 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that M' has index m and that P is a cover set for M. Let $Z \subseteq X^\star$ be nonempty and such that Z partitions the states of M' in at least m equivalence classes. If $s_0 \approx_{PZ} s'_0$ and $s_0 \not\approx s'_0$, then there exist $\gamma \in X^\star$, $s \in S$, $s' \in S'$ such that $\widehat{\delta}(\gamma, s_0) = s$, $\widehat{\delta'}(\gamma, s'_0) = s'$ and $s \not\approx_Z s'$.*

Proof Since $s_0 \not\approx s'_0$ we obtain $\gamma \in X^\star$, $a \in X$, $s \in S$ e $s' \in S'$ such that $\widehat{\delta}(\gamma, s_0) = s$, $\widehat{\delta'}(\gamma, s'_0) = s'$ and $\lambda(a, s) \neq \lambda'(a, s')$. Since P is a cover set for M, from edge $\delta(a, s) = r$ we obtain $\rho \in P$ e $\rho a \in P$ such that $\widehat{\delta}(\rho, s_0) = s$. We also know there exists $s'' \in S'$ such that $\widehat{\delta'}(\rho, s'_0) = s''$.

We show that $s \approx_Z s''$. Otherwise we would have $\sigma \in Z$ with $\widehat{\lambda}(\sigma, s) \neq \widehat{\lambda'}(\sigma, s'')$. Hence, $\widehat{\lambda}(\rho\sigma, s_0) = \widehat{\lambda}(\rho, s_0)\widehat{\lambda}(\sigma, s)$ and $\widehat{\lambda'}(\rho\sigma, s'_0) = \widehat{\lambda'}(\rho, s'_0)\widehat{\lambda'}(\sigma, s'')$ and we would obtain $\widehat{\lambda}(\rho\sigma, s_0) \neq \widehat{\lambda'}(\rho\sigma, s'_0)$, contradicting $s_0 \approx_{PZ} s'_0$ since $\rho\sigma \in PZ$. We conclude $s \approx_Z s''$.

Now we show that $s \not\approx_Z s'$. Otherwise we would have $s' \approx_Z s''$, since we already know that $s \approx_Z s''$. Since M' has index m and Z partitions the states of M' in m equivalence classes, we would obtain $s' \approx s''$. But then,

$$\begin{aligned}
\widehat{\lambda}(\rho a, s_0) &= \widehat{\lambda}(\rho, s_0)\lambda(a, s) \\
\widehat{\lambda'}(\rho a, s'_0) &= \widehat{\lambda'}(\rho, s'_0)\lambda(a, s'') = \widehat{\lambda'}(\rho, s'_0)\lambda(a, s').
\end{aligned}$$

Since we already have $\lambda(a, s) \neq \lambda'(a, s')$, we deduce $\widehat{\lambda}(\rho a, s_0) \neq \widehat{\lambda'}(\rho a, s'_0)$, which implies that $s_0 \not\approx_P s'_0$ because $\rho a \in P$. Since Z is nonempty, Lemma 10(3) forces $s_0 \not\approx_{PZ} s'_0$, contradicting the hypothesis. We conclude that $s \not\approx_Z s'$.

Putting together, we have $\gamma \in X^\star$, with $\widehat{\delta}(\gamma, s_0) = s$, $\widehat{\delta'}(\gamma, s'_0) = s'$ e $s \not\approx_Z s'$. ∎

Now we are in a position to enunciate the result which will give us the capability of testing two machines for equivalence.

Theorem 19 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that M' has index m and that P is a cover set for M. Let $Z \subseteq X^\star$ be nonempty and such that Z partitions states of M' in at least m equivalence classes. Then, $s_0 \approx s'_0$ if and only if $s_0 \approx_{PZ} s'_0$.*

Proof If $s_0 \approx s'_0$ then, trivially, $s_0 \approx_{PZ} s'_0$.

For the opposite direction, assume $s_0 \approx_{PZ} s'_0$. For the sake of contradiction, assume $s_0 \not\approx s'_0$. From Lemma 18, we obtain $\beta \in X^\star$, $s \in S$ and $s' \in S'$ with $\widehat{\delta}(\beta, s_0) = s$, $\widehat{\delta}'(\beta, s'_0) = s'$, and $s \not\approx_Z s'$. We can assume, without loss of generality, that $|\beta|$ is minimal. If $\beta = \epsilon$, we would have $s = s_0$ and $s' = s'_0$, and then $s_0 \not\approx_Z s'_0$. But, since $\epsilon \in P$, this would force the contradiction $s_0 \not\approx_{PZ} s'_0$. We conclude that $\beta = \alpha a$, with $a \in X$. Let $r \in S$ and $r' \in S'$ with $\widehat{\delta}(\alpha, s_0) = r$, $\widehat{\delta}'(\alpha, s'_0) = r'$, $\delta(a, r) = s$ and $\delta'(a, r') = s'$. Using the minimality of $|\beta|$ we have $r \approx_Z r'$.

On the other hand, since P is a cover set for M, from the edge $\delta(a, r) = s$ we obtain $\rho \in P$ and $\rho a \in P$ with $\widehat{\delta}(\rho, s_0) = r$. Let $r'' \in S'$ with $\widehat{\delta}'(\rho, s'_0) = r''$. If we had $r \not\approx_Z r''$, we would obtain $\gamma \in Z$ with $\widehat{\lambda}(\gamma, r) \neq \widehat{\lambda}'(\gamma, r'')$. But then

$$\begin{aligned} \widehat{\lambda}(\rho\gamma, s_0) &= \widehat{\lambda}(\rho, s_0)\widehat{\lambda}(\gamma, r) \quad \text{and} \\ \widehat{\lambda}'(\rho\gamma, s'_0) &= \widehat{\lambda}'(\rho, s'_0)\widehat{\lambda}'(\gamma, r''). \end{aligned}$$

Hence, $\widehat{\lambda}(\rho\gamma, s_0) \neq \widehat{\lambda}'(\rho\gamma, s'_0)$, giving the contradiction $s_0 \not\approx_{\rho\gamma} s'_0$ with $\rho\gamma \in PZ$. We conclude that $r \approx_Z r''$.

Since we already have $r \approx_Z r'$, we obtain $r' \approx_Z r''$. Since Z partitions the states of M' in m classes and m is the index of M', we conclude that $r' \approx r''$. Now, from $s \not\approx_Z s'$, we obtain $\sigma \in Z$ with $\widehat{\lambda}(\sigma, s) \neq \widehat{\lambda}'(\sigma, s')$. But,

$$\begin{aligned} \widehat{\lambda}(\rho a\sigma, s_0) &= \widehat{\lambda}(\rho, s_0)\lambda(a, r)\widehat{\lambda}(\sigma, s) \quad \text{and} \\ \widehat{\lambda}'(\rho a\sigma, s'_0) &= \widehat{\lambda}'(\rho, s'_0)\widehat{\lambda}'(a\sigma, r'') \\ &= \widehat{\lambda}'(\rho, s'_0)\widehat{\lambda}'(a\sigma, r') \\ &= \widehat{\lambda}'(\rho, s'_0)\lambda'(a, r')\widehat{\lambda}'(\sigma, s'). \end{aligned}$$

Then, $\widehat{\lambda}(\rho a\sigma, s_0) \neq \widehat{\lambda}'(\rho a\sigma, s'_0)$. But $\rho a\sigma \in PZ$ and we would have $s_0 \not\approx_{PZ} s'_0$, contradicting the hypothesis. This concludes the proof. ▮

Combining the previous results, we have the following corollary, useful to determine whether two FSMs have distinguishing behaviors.

Corollary 20 *Let M and M' two FSMs operating over the same input alphabet, X. Assume that M' has index m. Assume also that P is a cover set for M, that $R \subseteq X^\star$ is nonempty and that it partitions the states of M' in at least $n \leq m$ equivalence classes. Then, s_0 and s'_0 are equivalent if and only if s_0 and s'_0 are PZ-equivalent, where $Z = \bigcup_{i=0}^{m-n} X^i R$.*

Proof Let $Z_k = \bigcup_{i=0}^{k} X^i R$, $k \geq 0$. From Proposition 12 we have that such family $\{Z_k\}_{k\geq 0}$ is stratified. From Corollary 17 we conclude that Z distinguishes every pair of nonequivalent states of M'. Then the result follows directly from Theorem 19. ▮

5 The Concept of Characterization

In this section we revisit the notion of characterization sets, and following we show the relation of FSM indexes to such sets.

5.1 Characterization sets

From the previous Corollary 20, it might appear that Z and M are independent, since the only hypothesis involving M, in that corollary, is that P is a cover set for M. But, in fact, there is a relationship between Z and M. Before we expose the relationship between Z and M, we need another auxiliary result.

Lemma 21 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that all states of M are reachable and that $s_0 \approx s'_0$. Let $Z \subseteq X^\star$ be a set partitioning the states of M' in m equivalence classes, where m is the index of M'. Then Z distinguishes every pair of nonequivalent states of M.*

Proof Let $s_1, s_2 \in S$ with $s_1 \not\approx s_2$ and assume $s_1 \approx_Z s_2$. Since all states of M are reachable, we have $\rho_1, \rho_2 \in X^\star$ such that $\widehat{\delta}(\rho_i, s_0) = s_i$, with $i = 1, 2$. In M' we would have some $s'_1, s'_2 \in S'$ and with $\widehat{\delta}'(\rho_i, s'_0) = s'_i$, where $i = 1, 2$.

Now let $\beta \in Z$. We have,

$$\begin{aligned}\widehat{\lambda}(\rho_2\beta, s_0) &= \widehat{\lambda}(\rho_2, s_0)\widehat{\lambda}(\beta, s_2)\\ \widehat{\lambda}'(\rho_2\beta, s'_0) &= \widehat{\lambda}'(\rho_2, s'_0)\widehat{\lambda}'(\beta, s'_2)\end{aligned}$$

and, since $s_0 \approx s'_0$, we obtain $\widehat{\lambda}(\beta, s_2) = \widehat{\lambda}'(\beta, s'_2)$ and $\widehat{\lambda}(\rho_2, s_0) = \widehat{\lambda}'(\rho_2, s'_0)$. Since β is arbitrary, we conclude that $s_2 \approx_Z s'_2$.

Similarly,

$$\begin{aligned}\widehat{\lambda}(\rho_1\beta, s_0) &= \widehat{\lambda}(\rho_1, s_0)\widehat{\lambda}(\beta, s_1)\\ \widehat{\lambda}'(\rho_1\beta, s'_0) &= \widehat{\lambda}'(\rho_1, s'_0)\widehat{\lambda}'(\beta, s'_1),\end{aligned}$$

and we conclude that $s_1 \approx_Z s'_1$, together with $\widehat{\lambda}(\rho_1, s_0) = \widehat{\lambda}'(\rho_1, s'_0)$.

Putting it together, and knowing that $s_1 \approx_Z s_2$, we obtain $s_1 \approx_Z s'_2$ and also $s_2 \approx_Z s'_1$. Hence, $s'_1 \approx_Z s'_2$. But s'_1 and s'_2 are states of M' and so the hypothesis over Z gives $s'_1 \approx s'_2$.

On the other hand, since $s_1 \not\approx s_2$, we obtain $\sigma \in X^\star$ such that $\widehat{\lambda}(\sigma, s_1) \neq \widehat{\lambda}(\sigma, s_2)$. Now,

$$\begin{aligned}\widehat{\lambda}(\rho_1\sigma, s_0) &= \widehat{\lambda}(\rho_1, s_0)\widehat{\lambda}(\sigma, s_1)\\ \widehat{\lambda}'(\rho_1\sigma, s'_0) &= \widehat{\lambda}'(\rho_1, s'_0)\widehat{\lambda}'(\sigma, s'_1).\end{aligned}$$

Hence, from $\widehat{\lambda}(\rho_1\sigma, s_0) = \widehat{\lambda}'(\rho_1\sigma, s'_0)$ and $\widehat{\lambda}(\rho_1, s_0) = \widehat{\lambda}'(\rho_1, s'_0)$, we deduce $\widehat{\lambda}(\sigma, s_1) = \widehat{\lambda}'(\sigma, s'_1)$.

Similarly,

$$\begin{aligned}\widehat{\lambda}(\rho_2\sigma, s_0) &= \widehat{\lambda}(\rho_2, s_0)\widehat{\lambda}(\sigma, s_2)\\ \widehat{\lambda}'(\rho_2\sigma, s'_0) &= \widehat{\lambda}'(\rho_2, s'_0)\widehat{\lambda}'(\sigma, s'_2),\end{aligned}$$

and then $\widehat{\lambda}(\sigma, s_2) = \widehat{\lambda}'(\sigma, s'_2)$. However, since we already know that $s'_1 \approx s'_2$, this leads to the contradiction $\widehat{\lambda}(\sigma, s_1) = \widehat{\lambda}(\sigma, s_2)$. This shows that the initial hypothesis was false. Hence, whenever $s_1 \not\approx s_2$ holds we must also have $s_1 \not\approx_Z s_2$, establishing the result. ∎

A set in these conditions is called a characterization set of M.

Definition 22 *Let M be a FSM and W a set of input sequences. W is a characterization set for M if W distinguishes any pair of nonequivalent states of M.* ▮

The required relation between M and Z says that Z is a characterization set of M, under certain hypothesis.

Theorem 23 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that M' has index m and that P is a cover set for M. Assume also that $W \subseteq X^\star$ is nonempty and partitions the states of M' in at least $n \leq m$ equivalence classes. If $s_0 \approx_{PZ} s'_0$ then $Z = \bigcup_{i=0}^{m-n} X^i W$ is a characterization set for M.*

Proof From Proposition 12 and from Corollary 17 we conclude that Z distinguishes every pair of nonequivalent states of M'. Since P is cover set for M, we conclude that every state of M is reachable. From $s_0 \approx_{PZ} s'_0$, together with Corollary 20, we deduce $s_0 \approx s'_0$. Now we can use Lemma 21 and obtain that Z distinguishes every pair of nonequivalent states of M. From Definition 22, Z is a characterization set for M. ▮

It is also easy to see that the reverse does not hold. For that, let M and M' be two FSMs. It is clear that $W = X^\star$ partitions the states of M and M' in the maximum number of equivalence classes. In this case, we will have $Z = W = X^\star$ and, obviously, Z is a characterization set for M and M'. But it is not the case that we will always have $s_0 \approx s'_0$, as it is easy to construct a counter-example.

5.2 FSM Indexes

The index of FSMs has a strict relation with the notion of equivalence induced by characterization sets. Next result shows that, under relaxed conditions, when two FSMs are equivalent both must have the same index.

Theorem 24 *Let M and M' be two FSMs operating over the same input alphabet, X. Let n and n' be the index of M and M', respectively. Assume that all states from both FSMs are reachable. If $s_0 \approx s'_0$ then $n = n'$.*

Proof For the sake of contradiction, and without loss generality, we will assume $n < n'$.

Let $s'_i \in S'$, $1 \leq i \leq n'$, be states from each one of n' equivalence classes induced by $\approx$ in S'. Since all states of M' are reachable, we obtain $\rho_i \in X^\star$ with $\widehat{\delta}'(\rho_i, s'_0) = s'_i$, $1 \leq i \leq n'$. In M, we will have some $s_i \in S$ such that $\widehat{\delta}(\rho_i, s_0) = s_i$, $1 \leq i \leq n'$. Since $n < n'$, without loss generality, we can say that $s_1 \approx s_2$.

Take any $z \in X^\star$. We have

$$\begin{aligned}\widehat{\lambda}(\rho_1 z, s_0) &= \widehat{\lambda}(\rho_1, s_0)\widehat{\lambda}(z, s_1) \quad \text{and} \\ \widehat{\lambda}'(\rho_1 z, s'_0) &= \widehat{\lambda}'(\rho_1, s'_0)\widehat{\lambda}'(z, s'_1).\end{aligned}$$

Since $s_0 \approx s'_0$, it follows that $\widehat{\lambda}(z, s_1) = \widehat{\lambda}'(z, s'_1)$. Similarly, $\widehat{\lambda}(z, s_2) = \widehat{\lambda}'(z, s'_2)$.

But since $s_1 \approx s_2$, we obtain $\widehat{\lambda}(z, s_1) = \widehat{\lambda}(z, s_2)$. Therefore, $\widehat{\lambda}'(z, s_1') = \widehat{\lambda}'(z, s_2')$. Since $z \in X^\star$ is arbitrary, we conclude that $s_1' \approx s_2'$, a contradiction given that s_1' and s_2' are in distinct classes in M'.

Hence, we must have $n \geq n'$. Similarly, $n' \geq n$, and then $n = n'$. ▮

The same result indicates that when the $\approx$ relation induces a different number of equivalence classes in two FSMs, these machines can not be equivalent to each other (under the weak hypothesis of Theorem 24). On the other hand, it is simple to obtain two nonequivalent FSMs, in a such way that the $\approx$ relation induces the same number of equivalence classes in both machines.

6 Refining W-method as a particular case

In this section we show how to refine the W-method from our proposed method. Considering the hypothesis of Theorem 23 we can show that W is a characterization set of M if n is the index of M and the behaviors of both machines must match. From that we have conditions to show that W-method is a particular case of the generalization.

Corollary 25 *Let M and M' be two FSMs operating over the same input alphabet, X, and assume that all states in M' are reachable. Assume further that M' has index m, that P is a cover set for M and that M has index n. Assume also that $W \subseteq X^\star$ is nonempty and partitions the states of M' in at least $n \leq m$ equivalence classes. If $s_0 \approx_{PZ} s_0'$, where $Z = \bigcup_{i=0}^{m-n} X^i W$, then $n = m$, $Z = W$ and W is a characterization set for M.*

Proof Since $s_0 \approx_{PZ} s_0'$, together with Corollary 20, we conclude that $s_0 \approx s_0'$. Next, we infer that $n = m$, from Theorem 24. Hence, $Z = W$. Therefore, by Theorem 23, W is a characterization set for M. ▮

When W is a characterization set for M we can guarantee the partitioning of M' in a number of classes at least equal to the index of M, if the machines are to be PZ-equivalent.

Lemma 26 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that M' has index m, that M has index n and that P is a cover set for M, with $n \leq m$. Assume also that $W \subseteq X^\star$ is a characterization set for M and that $s_0 \approx_{PZ} s_0'$, where $Z = \bigcup_{i=0}^{m-n} X^i W$. Then W partitions M' in at least n equivalence classes.*

Proof We know that M has n equivalence classes: Let $C_1, \ldots, C_n$ be these classes. Let $s_i \in C_i$ and $s_j \in C_j$, where $1 \leq i < j \leq n$. Then since W is a characterization set for M, we have $s_i \not\approx_W s_j$. Since P is cover set of M, we have $\widehat{\delta}(\rho, s_0) = s_i$, for some $\rho \in P$. We also know that $\widehat{\delta}'(\rho, s_0') = s_i'$, for some s_i' of M'. Since $s_0 \approx_{PZ} s_0'$, we get $s_i \approx_Z s_i'$. Since $W \subseteq Z$, then $s_i \approx_W s_i'$. In the same way, we have s_j' of M' with $s_j \approx_W s_j'$. Then we obtain $s_i' \not\approx_W s_j'$, otherwise $s_i \approx_W s_j$. We conclude that W partitions M' in at least $n \leq m$ equivalence classes. ▮

Now we can use Lemma 26 to show another version of Corollary 20, under the hypothesis that the basic set of input sequences is a characterization set for the specification.

Theorem 27 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that M' has index m, that P is a cover set for M and that M has index n, with $n \leq m$. Assume also that $W \subseteq X^\star$ is a characterization set for M and that $s_0 \approx_{PZ} s'_0$, where $Z = \bigcup_{i=0}^{m-n} X^i W$. Then $s_0 \approx s'_0$.*

Proof Assume $s_0 \approx_{PZ} s'_0$. Use Lemma 26 to show that W partitions M' in at least n classes. Now use Corollary 17 to show that Z partitions M' in m classes. Finally, use Theorem 19. ▮

The next result is the main postulate of the basic W-method, as given in [3].

Theorem 28 *Let M and M' be two FSMs operating over the same input alphabet, X. Assume that M' has index m, that P is a cover set for M and that M has index n, with $n \leq m$. Assume also that $W \subseteq X^\star$ is a characterization set for M. Then $s_0 \approx s'_0$ if and only if $s_0 \approx_{PZ} s'_0$, where $Z = \bigcup_{i=0}^{m-n} X^i W$.*

Proof If $s_0 \approx s'_0$, then $s_0 \approx_{PZ} s'_0$, trivially. For the other direction, use Theorem 27. ▮

In general, W does not need to be a characterization set for M (see Corollary 20). For the method to work, we need only guarantee that M' will be partitioned in at least n equivalence classes with $n \leq m$, where m is the index of M'. No relationship between W and M is needed. On the other hand, when using the basic W-method directly, we need to obtain a characterization set W for M, we need to know the index of M, and we also need to secure the relationship $n \leq m$. When W is not a characterization set for M, the method may fail, as shown by the following example.

Example 29 Let M and M' be two FSMs. The alphabet of M and M' is $X = \{a, b, c\}$. See Figures 1 and 2. It is easy to see that M has index $n = 3$. The index of M' is $m = 3$ since $s'_1 \approx s'_3$. Hence $m = n$, and we would be left with $Z = W$ (see Theorem 28). Now take $W = \{a\}$. A cover set can be given by $P = \{\epsilon, aa, ab, ac, ba, bb, bc, ca, cb, cc\}$. Then, $PZ = PW = \{aaa, aba, aca, baa, bba, bca, caa, cba, cca\}$.

It is easy to see that M and M' are PZ-equivalent. But $s_0 \approx s'_0$ is not true. To see that, take $\alpha = bab$. We have $\widehat{\lambda}(\alpha, s_0) = 110$ and $\widehat{\lambda}'(\alpha, s'_0) = 111$. Note how W induces only two equivalence classes in M'. Therefore, clearly, W is not a characterization set for M. ▮

In general, it would be important to devise a mechanism by which we could obtain the number of classes induced by W in M'. First, because in this case we might avoid calculating a characterization set for M when using our more general method. Secondly, we could potentially reduce the size of the sequences in Z, when W partitions M' in k classes, with $k > n$, given that $Z = \bigcup_{i=0}^{m-n} X^i W$.

7 The Generalized Test Generation Method

In this section we present Algorithm 1 of the generalized model-based test generation method. The input parameters are: M represents a system specification, R is any set

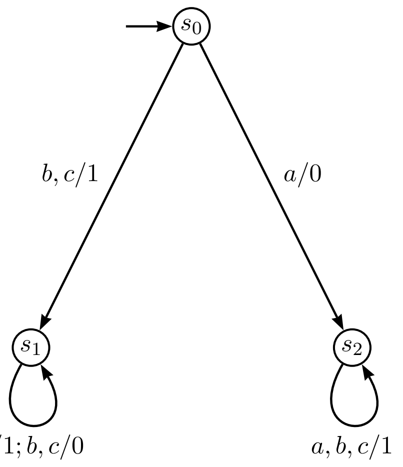

Figure 1: Specification M.

of input sequences, n is a lower bound on the number of classes induced by R in an implementation candidate, and m is an upper bound on the index of implementation candidates. Thus the method requires knowledge of a lower bound on the number n of equivalence classes induced by R in the implementation candidate, as well as an upper bound on the index m of such implementation candidates. In an extreme case, one can set $n = 1$ and $m = |S'|$, that is, set m to the number of states in implementation candidates. Note that an implementation candidate is given as a black box. So, we do not have access to its internal structure, and the parameters n and m must be estimated. As for the specification M, R may partition it in any number k of classes. Of course, if M and an implementation candidate turn out to be equivalent, then they will have the same index and Z will, in fact, be a characterization set for both M and the implementation candidate as seen before. The result is the set of test sequences given by PZ which can be empty if the basic condition $n \leq m$ is not secured.

After the Algorithm 2 presents the application of the generalized method. The input parameters are: M represents a system specification, M' is an implementation candidate for the specification M, and PZ is the set of test sequences. The result is the verdict if M and M' are not equivalent, then the algorithm produces a particular input sequence σ that is a witness to this fact, that is, M and M' display distinct behaviors over σ. On the other hand, M and M' are equivalent.

In order to apply the basic W-method (see Theorem 28) some extra effort must be applied to compute the index of M as well as a characterization set for M. Algorithm 3 presents the basic W-method for the test case generation. The application is similar to the Algorithm 2.

Note that in our proposal, we do not need characterization sets, nor is it necessary to inform the index of the specification machine M. On the other hand, practical information about M can aid in obtaining a good candidate for R. For example, based on the number of symbols in the input alphabet and on the number of states and transitions in M, some

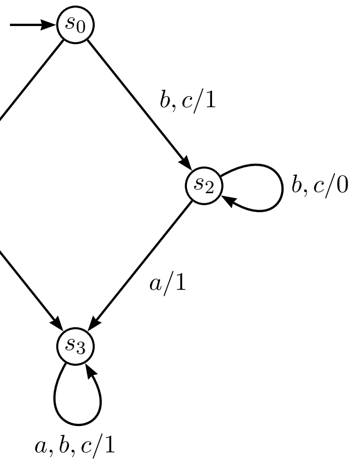

Figure 2: Implementation candidate M'.

distinguishing sequences can be inserted into R (see Section 8). Then, it is easy to obtain the set Z using the notion of stratification. Clearly, after obtaining the concatenation PZ, we can use this product to verify conformance between the specification and several proposed implementations.

Note that the size of the PZ set depends on the algorithm used to obtain the cover set P. In fact, this algorithm is polynomial in the size of M (see Section 2.2). Furthermore, it depends on the choice of the set R and the bound m.

Next, we apply the generalized algorithm to a simple and general example.

Example 30 Let M be a specification given as in Figure 3. Then M has $k = 4$ states, its input alphabet is $X = \{a, b\}$, its output alphabet is $Y = \{0, 1\}$, and its transition function is as depicted in the figure.

As we can see, some transitions over the input a produce either the output 0 or the output 1. Hence, there are at least two distinct classes. Now, if we use the sequences aa and ba there is a good chance that such sequences can distinguish other states as well. Therefore, we take $R = \{aa, ba\}$ and assume that R partitions M', an implementation candidate, in at least $n = 3$ equivalence classes. If we accept $m = 5$ as a maximum on the number of states in M', we have all input conditions for Algorithm 1 secured. Note that R is not a characterization set for M because we have states s_0 and s_1 in the same equivalence class induced by R.

Next we calculate a cover set P for M. In the example, using the labeled tree construction (see Section 2.2) we get $P = \{\epsilon, a, b, aa, ab, ba, bb, aab, aaa\}$.

Now, with $m = 5$, $n = 3$ and $R = \{aa, ba\}$, we compute $Z = \bigcup_{i=0}^{m-n} X^i R$ and obtain

$$Z = \{aa, ba, aaa, aba, baa, bba, aaaa, abaa, baaa, bbaa, aaba, abba, baba, bbba\}.$$

Taking Z as a prefix-free (see Section 8), and also for the concatenation PZ, then PZ will count 40 sequences.

Algorithm 1: Generalized test generation method.

```
1  Input: $M$, $R$, $m$, $n$
2  Output: $PZ$
3  begin
4      Obtain a cover set $P$ for $M$;
5      if $n \leq m$ then
6          Compute $Z = \bigcup_{i=0}^{m-n} X^i R$;
7          Compute $PZ$;
8          return $PZ$;
9      else
10         mesg: $M$ and $M'$ could not be equivalent since $m < n$;
11         return $PZ = \emptyset$;
12     end
13 end
```

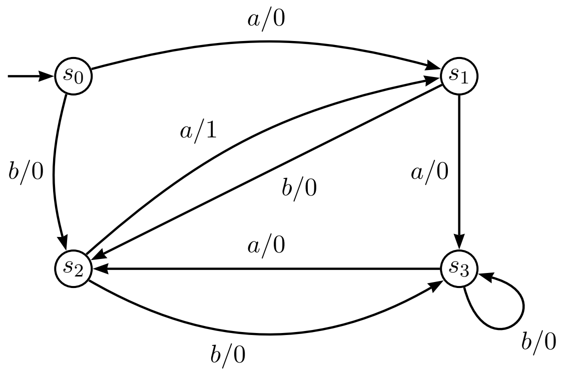

Figure 3: Machine specification M.

8 Special Families of FSMs

In this section we compare the W-method against the G-method over some infinite families of models. We show that there exist infinite families of FSMs such that the application of the G-method produces test suites of much smaller length than those produced by the W-method.

Consider an alphabet X and a test suite $V \subseteq X^\star$. Testing using V would require observing the behavior of both the specification and the implementation models over all elements of V. Therefore, it is not necessary to consider any proper prefix in V, since testing over a longer sequence will already reveal behaviors over any shorter prefix.

Definition 31 *Let X be an alphabet and let $V \subseteq X^\star$. Define* pff(V) *as the set of prefix-free elements of V, i.e.,*

$$\text{pff}(V) = \{\varphi \in V \mid \varphi\vartheta \notin V \textit{ for all } \vartheta \in X^\star \textit{ with } \vartheta \neq \varepsilon\}.$$

Algorithm 2: The application of the generalized test generation method.

```
Input: $M$, $M'$, $PZ$
Output: $\sigma$
begin
    foreach $\sigma \in PZ$ do
        Apply $\sigma$ to $M$ and to $M'$;
        Obtain $\alpha = \widehat{\lambda}(\sigma, s_0)$ and $\alpha' = \widehat{\lambda}'(\sigma, s'_0)$;
        if $\alpha \neq \alpha'$ then
            mesg: $M$ and $M'$ are not equivalent;
            return $\sigma$ is an input witness;
        end
    end
    mesg: $M$ and $M'$ are equivalent;
    return $\sigma = \epsilon$;
end
```

The efficiency of a test suite $V \subseteq X^\star$ will be measured as the sum of the lengths of all elements in $\mathrm{pff}(V)$.

Definition 32 *Let X be an alphabet and let $V \subseteq X^\star$. We define*

$$\|V\| = \sum_{\sigma \in \mathrm{pff}(V)} |\sigma|.$$

The next result is useful in calculations.

Lemma 33 *Let $V_1, V_2 \subseteq X^\star$, with V_2 finite. Then* $\mathrm{pff}(V_1 V_2) = \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$.

Proof First, let $\sigma \in \mathrm{pff}(V_1V_2)$. Then $\sigma = \alpha_1\alpha_2$ with $\alpha_1 \in V_1$ and $\alpha_2 \in V_2$. If $\alpha_2 \notin \mathrm{pff}(V_2)$, then $\alpha_2\beta \in V_2$ with $\beta \neq \epsilon$. But this gives $\sigma\beta \in V_1V_2$, contradicting $\sigma \in \mathrm{pff}(V_1V_2)$. Thus, $\alpha_2 \in \mathrm{pff}(V_2)$, and so $\sigma \in V_1 \cdot \mathrm{pff}(V_2)$. If $\sigma \notin \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$, then $\sigma\gamma \in V_1 \cdot \mathrm{pff}(V_2)$ with $\gamma \neq \epsilon$. But then, again, $\sigma\gamma \in V_1V_2$ contradicting $\sigma \in \mathrm{pff}(V_1V_2)$. Thus, $\sigma \in \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$ and we conclude that $\mathrm{pff}(V_1V_2) \subseteq \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$.

Now, assume that we have $\sigma \in X^\star$ with $\sigma \in \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$ and $\sigma \notin \mathrm{pff}(V_1V_2)$. Then, $\sigma \in V_1 \cdot \mathrm{pff}(V_2)$ and so $\sigma = \alpha_1\alpha_2$, with $\alpha_1 \in V_1$ and $\alpha_2 \in \mathrm{pff}(V_2)$. We get $\alpha_2 \in V_2$ and $\sigma \in V_1V_2$. Clearly, the hypothesis gives some $\beta \in X^\star$ such that $\sigma\beta \in V_1V_2$ with $\beta \neq \epsilon$. Therefore, $\alpha_1\alpha_2\beta \in V_1V_2$ and we may write $\alpha_1\alpha_2\beta = \rho_1\rho_2$ with $\rho_1 \in V_1$ and $\rho_2 \in V_2$. Hence, there are two cases:
Case 1: $\rho_2 \in \mathrm{pff}(V_2)$. We obtain $\rho_1\rho_2 \in V_1 \cdot \mathrm{pff}(V_2)$ and so $\alpha_1\alpha_2\beta \in V_1 \cdot \mathrm{pff}(V_2)$. But then $\sigma\beta \in V_1 \cdot \mathrm{pff}(V_2)$, contradicting $\sigma \in \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$.
Case 2: $\rho_2 \notin \mathrm{pff}(V_2)$. Then $\rho_2 \in V_2$ and, since V_2 is finite, we get some longest $\gamma \in X^\star$ such that $\rho_2\gamma \in \mathrm{pff}(V_2)$. Again, $\rho_1\rho_2\gamma \in V_1 \cdot \mathrm{pff}(V_2)$. But now $\rho_1\rho_2\gamma = \alpha_1\alpha_2\beta\gamma = \sigma\beta\gamma$, contradicting $\sigma \in \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$. Therefore, if $\sigma \in \mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2))$ we must have $\sigma \in \mathrm{pff}(V_1V_2)$, showing that $\mathrm{pff}(V_1 \cdot \mathrm{pff}(V_2)) \subseteq \mathrm{pff}(V_1V_2)$.

Algorithm 3: The basic W-method.

```
Input: $M$, $M'$, $m$
Output: $PZ$
begin
    Obtain a cover set $P$;
    Obtain a characterization set $W$ for $M$;
    Obtain the index $n$ of $M$;
    if $n \leq m$ then
        Compute $Z = \bigcup_{i=0}^{m-n} X^i W$;
        Compute $PZ$;
        return $PZ$;
    else
        mesg: $M$ and $M'$ could not be equivalent since $m < n$;
        return $PZ = \emptyset$;
    end
end
```

We conclude that $\text{pff}(V_1V_2) = \text{pff}(V_1 \cdot \text{pff}(V_2))$, as desired. ∎

Clearly, all practical test suites are finite, thus satisfying the conditions of lemma and allowing one to compute $\text{pff}(V_1V_2)$ by calculating $\text{pff}(V_1 \cdot \text{pff}(V_2))$ which is potentially simpler. That notwithstanding, the next example shows that the finitude condition on V_2 can not be relaxed.

Example 34 *Let* $V_1 = \{ab, a\}$ *and let* $V_2 = \{c\} \cup \{bca^i \,|\, i \geq 0\}$.

Then $V_1V_2 = \{ac, abc\} \cup \{ab^2ca^i, abca^i \,|\, i \geq 0\}$, *and we get* $\text{pff}(V_1V_2) = \{ac\}$. *Also,* $\text{pff}(V_2) = \{c\}$ *and so* $V_1 \cdot \text{pff}(V_2) = \{abc, ac\} = \text{pff}(V_1 \cdot \text{pff}(V_2))$.

Taking $\sigma = abc$ *we get* $\sigma \in \text{pff}(V_1 \cdot \text{pff}(V_2))$ *and* $\sigma \notin \text{pff}(V_1V_2)$. *This shows that* $\text{pff}(V_1 \cdot \text{pff}(V_2)) \not\subseteq \text{pff}(V_1V_2)$ *when* V_2 *is not finite.* ∎

Now, consider the specification model depicted in Figure 4, where $n \geq 0$ and $X = \{a, b\}$. Clearly, the machine has $n+1$ states. Let $\rho = a^{n+1}$. We get $\widehat{\lambda}(\rho, s_i) = 0^{n-i}1^{i+1}$, for all i, $0 \leq i \leq n$. Hence, $\widehat{\lambda}(\rho, s_i) \neq \widehat{\lambda}(\rho, s_j)$ if $i \neq j$, and we may conclude that $s_i \not\approx s_j$ in this model, for all i, j, $i \neq j$, and $0 \leq i, j \leq n$. This shows that M_n is a minimal FSM with index $n+1$.

We may take $W = \{a^{n+1}\}$ as a characterization set for M_n. Note that, in addition, $\|W\|$ is minimal among all characterization sets for M_n. A simple application of Construction 3 (see subsection 2.2) gives a transition cover for M_n:

$$P = \{\epsilon\} \cup \{a^i a, a^i b \,|\, 0 \leq i \leq n\}.$$

The implementation model, M'_n, is obtained by changing the output of just one transition at the last state in M_n. See Figure 5. We take $m + 1 = \lceil \alpha \cdot (n+1) \rceil$, for some parameter

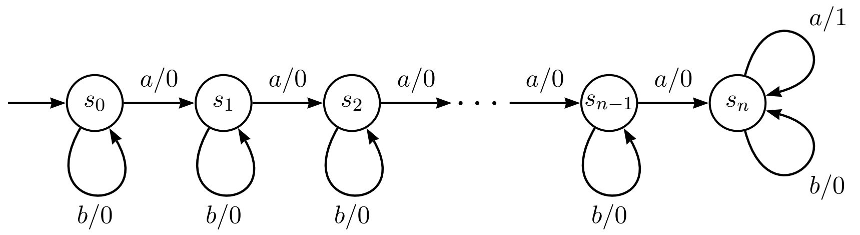

Figure 4: Specification model M_n.

$\alpha > 1$. That is, the implementation model has up to $100(\alpha - 1)\%$ more states than the specification M_n. Then, if we want to test implementations with up to 5% more states, we let $\alpha = 1.05$. As before, one can easily see that M'_n is a minimal FSM with index $m + 1$.

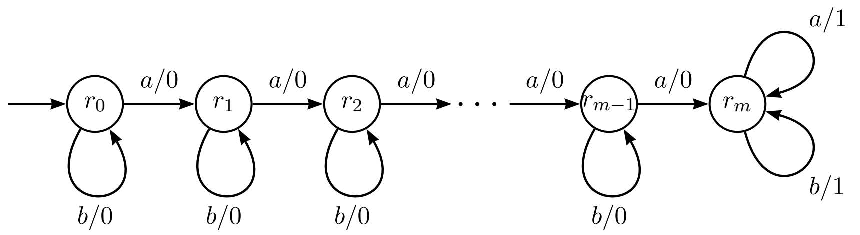

Figure 5: Implementation model M'_n.

Consider the set of input sequences $R = \{a^m b\}$. Then $\widehat{\lambda}'(a^m b, r_j) = 0^{m-j}1^{j+1}$, for all j, $0 \le j \le m$. Hence, $r_i \not\approx_R r_j$ in the implementation model M'_n for all i, j with $i \ne j$ and $0 \le i, j \le m$. Thus, R partitions the states of M'_n in $k = m + 1$ equivalence classes.

Now, we can compare the test suites that are obtained using the G-method and the W-method. We indicate the Z sets constructed by the G-method and the W-method by Z_R and Z_W, respectively. We will show that the ratio $\frac{\|PZ_W\|}{\|PZ_R\|}$ grows exponentially fast, as n increases[1].

We get

$$Z_R = \bigcup_{i=0}^{(m+1)-k} X^i R = \{\epsilon\} \cdot R = \{a^m b\}$$

$$PZ_R = \big(\{\epsilon\} \cup \{a^i a, a^i b \,|\, 0 \le i \le n\}\big) \cdot \{a^m b\} = \{a^m b\} \cup \{a^i a a^m b, a^i b a^m b \,|\, 0 \le i \le n\}.$$

Therefore, even counting all sequences in PZ_R, we have

$$\|PZ_R\| \le (m+1) + 2(m+2)\sum_{i=0}^{m} i = (m+1)^3 \le (\alpha(n+1)+1)^3 \le 8\alpha^3 n^3,$$

[1]We use the standard big-Oh and big-Omega notation from complexity theory [21].

assuming $n \geq 2$ and noting that $\alpha > 1$.

Over the specification model M_n we compute

$$Z_W = \bigcup_{i=0}^{(m+1)-(n+1)} X^i W = \bigcup_{i=0}^{\ell} X^i \{a^{n+1}\},$$

where $\ell = \lceil (n+1)(\alpha - 1) \rceil$. So, all sequences in the form $a^n b\, \rho\, a^{n+1}$ are in pff(PZ_W), for all $\rho \in X^\ell$. Note that $a^n b$ is in P and has maximum length among all such sequences. Thus, $a^n b\, \rho\, a^{n+1}$ has maximum length among all sequences in PZ_W. Hence, we may write $\|PZ_W\| \geq (2n + 2 + \ell)2^\ell \geq n2^{n(\alpha-1)}$.

Now, consider the ratio $Q_\alpha(n) = \frac{\|PZ_W\|}{\|PZ_R\|}$. Clearly, we have that $Q_\alpha(n)$ is $\Omega(2^{cn})$ for all c such that $n(\alpha - 1) - nc > 0$, that is, for all c such that $c < \alpha - 1$.

We conclude that the family of specification and implementation pairs M_n and M'_n, respectively, is such that the G-method generates test suites that are exponentially more effective than those generated using the W-method, provided that the fixed parameter α satisfies $\alpha > 1$.

Note that this condition can, potentially, arise when models M_n and M'_n occur as submodels in other specifications and implementations, respectively. Also, note that we can freely change the target of all b in the implementation M'_n and the result will still hold. That is, the G-method generates exponentially shorter test suites that can be used to test a whole class of implementations. Similarly, one can freely change the target of all b transitions in the specification model M_n and still obtain the same result.

This suggests an alternative testing strategy as follows. Using short effective test suites produced by the G-method, with a judicious choice of parameters, we can test a large number of models from the set of implementation candidates, assuming a fixed upper bound on the number of states for the implementation models. In case some implementation models pass this first test suite, one could produce more stringent test suites, and test the remaining implementation models in a second pass. For this second pass one could, with extra effort, construct complete test suites using the G-method with stronger parameters, using the original W-method, or any other complete method.

9 Related Works

In this section we summarize the W-method and briefly describe other model-based test generation methods, such as the W_p and the HSI methods.

9.1 The W-method

In this method, the aim is to verify whether an implementation conforms to a specification, as characterized by the behavior responses generated by external stimuli [3].

Basically, the application of this method consists in two steps, given a specification FSM M and an implementation FSM M': (i) test sequences generation, based on M; and (ii) application of each test sequence to M and M', followed by a comparison of their respective behaviors.

The technique uses characterization sets of M in order to obtain a complete set of test case sequences. A characterization set, loosely speaking, can distinguish every pair of machine states (see Section 5). Let W be a characterization set for M. In order to obtain test sequences, the W-method prefixes the sequences in W with certain sequences of input symbols, thus obtaining a set Z containing extended sequences. Furthermore, the method also computes a cover set P for M. A cover set, basically, contains sequences that traverse any edge of M, starting from the initial state. The desired set of test sequences is the product PZ.

9.2 The W_p-method

A related method, the so called W_p-method [6], can potentially reduce the total length of the test sequences generated by the basic W-method. Again, let W be a characterization set for the specification model, M. For each state s_i of M, a so called identification subset $W_i \subseteq W$ is obtained. The idea is that for each state s_j of M, with $s_i \neq s_j$, there exists an input sequence $\rho_j \in W_i$ such that s_i and s_j are ρ_j-distinguishable, and no other proper subset of W_i has this property.

Then, a checking sequence for each state is prefixed to all sequences in the corresponding identification set. A checking sequence for a given state is simply an input sequence that reaches that state, when starting at the initial state. It can be shown [6] that the length of the resulting test sequences may be shorter, when compared to those sequences obtained using the complete PZ concatenation set of the basic W-method.

9.3 The HSI-method

The HSI-method [16] uses the notion of trace-inclusion and a quasi-equivalence relation to verify conformance between partial non-deterministic FSM implementations and a given FSM specification. For that, so called harmonized sate identification sets are used instead of the identification subsets used in the W_p-method. Whereas identification sets fixed the sequences associated with a specific state s_i, a harmonized state identification set D_i, is constructed by taking prefixes of a characterization set W, but now allowing the reuse of a same prefix for different states. Distinguishing sequences for states are then taken from the intersection of D_i-sets. It is argued that shorter sequences can be found to distinguish every pair of states in M [16].

10 Concluding Remarks

The Finite State Machine (FSM) model is well established and has been intensively investigated as a foundation for the automatic generation of test cases. The W-method is a well known technique used to compute test sequences having FSMs as its basic formal model.

In this work, we generalize the basic W-method, thus obtaining the G-method, which altogether avoids the computation of characteristic sets and indexes for the specification models. We also demonstrated in a clear way how the basic W-method follows from the proposed G-method. We presented detailed proofs of correctness for both the main algorithm in the G-method, as well as for the main tenets of the basic W-method, the latter being absent in the original work where it was introduced.

We also showed some infinite families of FSM models where G-method obtains exponentially more succinct test case suites than does the W-method. This suggested alternative testing strategies, where a compact test suite, generated using the G-method, is first applied to the target implementations followed, if necessary, by a second step were more complete suites are constructed and subsequently applied to the remaining implementations.

Some more recent test generation methods, such as those presented in Section 9, also need to extract characterization sets and indices from the specification models, as in the basic W-method, in order to properly construct test suites. On the other hand, the G-method does not need such characterization sets or indices in order to generate test cases. We envisage that similar ideas can be used to extend and generalize other test case generation techniques, such as the W_p and HSI methods.

References

[1] Adilson Luiz Bonifácio, Arnaldo Vieira Moura, and Adenilso da Silva Simão. A generalized model-based test generation method. In Antonio Cerone and Stefan Gruner, editors, *Sixth IEEE International Conference on Software Engineering and Formal Methods, SEFM*, pages 139–148, Cape Town, South Africa, 10–14, nov 2008. IEEE Computer Society.

[2] Rachel Cardell-Oliver. Conformance tests for real-time systems with timed automata specifications. *Formal Aspects of Computing*, 12(5):350–371, 2000.

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

[4] Steven J. Cunning and Jerzy W. Rozenblit. Automating test generation for discrete event oriented embedded system s. *J. Intell. Robotics Syst.*, 41(2-3):87–112, 2005.

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

[6] 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.

[7] A. Gargantini. Conformance testing. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors, *Model-Based Testing of Reactive Systems: Advanced Lectures*, volume 3472 of *Lecture Notes in Computer Science*, pages 87–111. Springer-Verlag, 2005.

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

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

[10] F. C. Hennie. Fault detecting experiments for sequential circuits. In *FOCS*, pages 95–110, 1964.

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

[12] M. Krichen. State identification. In M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors, *Model-Based Testing of Reactive Systems: Advanced Lectures*, volume 3472 of *Lecture Notes in Computer Science*, pages 87–111. Springer-Verlag, 2005.

[13] M. Krichen and S. Tripakis. Black-box conformance testing for real-time systems. In *Model Checking Software: 11th International SPIN Workshop*, number 2989 in Lecture Notes in Computer Science, pages 109–126, Barcelona, Spain, April 2004.

[14] Gang Luo, G. von Bochmann, and A. Petrenko. Test selection based on communicating nondeterministic finite-state ma chines using a generalized wp-method. *IEEE Trans. Softw. Eng.*, 20(2):149–162, 1994.

[15] Brian Nielsen and Arne Skou. Test generation for time critical systems: Tool and case study. *ecrts*, 00:0155, 2001.

[16] Alexandre Petrenko and Gregor v. Bochmann. Selecting test sequences for partially-specified nondeterministic finite state machines. In Gang Luo, editor, *IWPTS '94: 7th IFIP WG 6.1 international workshop on Protocol test systems*, pages 95–110, London, UK, UK, 1995. Chapman & Hall, Ltd.

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

[18] Ali Rezaki and Hasan Ural. Construction of checking sequences based on characterization sets. *Computer Communications*, 18(12):911–920, 1995.

[19] D. Sidhu and T. Leung. Experience with test generation for real protocols. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 257–261, New York, NY, USA, 1988. ACM.

[20] Deepinder P. Sidhu and Ting kau Leung. Formal methods for protocol testing: A detailed study. *IEEE Trans. Softw. Eng.*, 15(4):413–426, 1989.

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

[22] Jan Tretmans. Test generation with inputs, outputs, and quiescence. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for Construction and*

Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings, volume 1055 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 1996.

[23] Jan Tretmans. Testing concurrent systems: A formal approach. In J.C.M Baeten and S. Mauw, editors, *CONCUR '99: Proceedings of the 10th International Conference on Co ncurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 46–65, London, UK, 1999. Springer-Verlag.

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