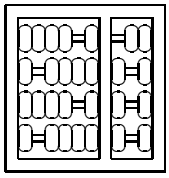

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Testing Combined Finite State Machines

L. L. C. Pedrosa *A. V. Moura*

Technical Report - IC-10-01 - Relatório Técnico

January - 2010 - Janeiro

Testing Combined Finite State Machines

Lehilton Lelis Chaves Pedrosa* Arnaldo Vieira Moura†

Abstract

The automatic generation of test cases is an important problem for conformance testing of several critical systems. In many situations, the system specification is modeled as a Finite State Machine (FSM). In practice, a system is a combination of several subsystems, designed, developed and tested independently. Since the overall number of states in the FSM specification is usually large, the known methods to generate test suites may become impractical. So, to properly describe modular systems, in this paper, we define the concept of combined FSMs. A combined FSM is obtained conjoining previously tested submachines with newly added states. We adapt the well-known W-method [4] and the G-method [2], and introduce a new method to test combined FSMs. The new method is scalable to the number of states, and can also be used for incremental testing of new systems, or for retesting modified implementations.

1 Introduction

Formal test case generation for reactive and critical systems using model based strategies has been widely studied [1, 2, 4, 6, 10, 13, 15, 19]. In such methods, the systems' requirements are described by means of mathematical models and formally specified functionalities. When applying testing strategies, the notion of conformance can be adopted, so that, if an implementation passes a test suite, its behavior is said to conform to the behavior extracted from the specification [5]. The methods that automate the generation of test suites, in general, are required to have two contrasting features [3, 13]:

1. They must be *efficient* in terms of test case length, so the smaller the test suite is, the faster the implementation can be tested.

2. They must be *accurate* in terms of fault coverage, so the test suite must detect as many implementation faults as possible.

Finite State Machines (FSMs) are used as the basic formalism in many methods that automate the generation of conformance test case suites. For surveys on this topic, see [1, 13, 19]. Among such methods, the so called W-method [4] is based on the notion of characterization sets, and provides full fault coverage for minimal, completely specified and

*Work supported by FAPESP grant 08/07969-9.
†Work supported by FAPESP grant 02/07473-7.

deterministic FSMs. Its main idea relies on testing whether the implementation has a corresponding state for each state in the specification. Several derivations have been proposed around it. In particular, the G-method [2] is a generalization of the W-method that does not depend on characterization sets. However, it still needs to check equivalence for each state in the specification, so it also relies on building a complete state cover set for the specification.

These methods assume that the system specification is an only block. However, in many situations, we find that systems are modular, with their specifications being formed by several subsystems. If one such subsystem is also modeled by a FSM, we call it a submachine. Then, the full FSM model can be obtained by conjoining several submachines, with the aid of a few new states and transitions. In this article, we propose a new approach to test combined FSMs, in cases when submachine implementations are known to be correct in advance.

When verifying whether an implementation conforms to a given specification, one needs to check the entire model. This means that we need to check whether there exists an equivalence between states in the implementation and in the specification. In practice, this is not always necessary, because a subset of states of the specification may already be known to have equivalent states in the implementation. This happens in at least two situations:

1. Creating a new FSM based on other existing and correctly implemented FSMs.

2. Modifying models, so that only some specific subset of the states are involved.

Consider the first case. Suppose that a system specification is a combination of several submachines, obtained by adding some new states and transitions. If these submachines are already correctly implemented, then the system implementation can integrate such submachines' implementations. Although each submachine implementation has already been tested, the resulting combination can still have an unexpected behavior, because the new implementation may not have correctly interconnected the submachines, or may have incorrectly added new states and transitions. One could test the implementation by applying some traditional test case generation methods, such as the W-method or the G-method. This would lead to accurate test suites, but, since these methods ignore the information about the submachines, they would be redundant, and thus inefficient. Now consider the other case. Systems evolve with time, so, if a given specification is changed, only the corresponding part of a former implementation gets modified. Again, full testing using methods like the W-method or the G-method would be inefficient.

Retesting modified implementations has already been studied. Koufareva et. al. [12] presented methods for restricted types of errors, and El-Fakih [8, 9] introduced methods for retesting an implementation with controlled types of modifications. In this paper, we do not restrict the types of errors the implementation can have, neither how it is modified. Our method is based on the main assumption that an implementation have some equivalent states in the specification. Additionally, we allow implementations with more states than in the specification.

In Section 2, we review the FSM model and present some conventions used in this paper. We also introduce state neighborhoods and notion of relative concatenation.

In Section 3, we describe equivalence relations between states of two FSMs. We also introduce the concept of separators, as a way to distinguish states of two given sets. Separators generalize the notion of characterization sets used in the W-method [4], and the idea of identification sets used in the Wp-method [10]. We use separators to distinguish the additional states in an implementation from states in the participating submachines. We also show that it is not necessary even to have complete implementation characterization sets, in order to identify states, in contrast to the W-method and the G-method.

In Section 4, we formalize the notion of a combined FSM, based on the practical hypothesis that systems are developed using the building block strategy. First, we precisely define a submachine. Then, we define models formed by conjoining one or more submachines.

In Section 5, we present a new test case generation method, here named the C-method. We first describe the fault model. We assume that the implementation is a deterministic and completely specified FSM, created by integrating previously tested submachine implementations with up to m *additional* states. Next, we present the test suite construction method. The generated test suite provides full fault coverage, and, although inspired by the W-method and the G-method, it requires only part of the cover set, and replaces the characterization set or the user provided sets by separators.

In Section 6, we compare our method with the W-method. We show that the size of a test suite generated by the C-method is exponential on the number of additional states, and polynomial on the number of submachine states. This implies that, unlike the W-method or the G-method, the C-method is scalable, that is, it can be used to test FSMs with a much larger number of states. Further, we show that the ratio between the number of test cases generated by the W-method and the number of test cases generated by the C-method is exponential on the number of submachine states. Finally, we give a simple practical example to illustrate C-method.

In Section 7, we prove the validity of the C-method.

In Section 8, we make some remarks on the new method.

2 Basic definitions

In this section, the FSM model is reviewed, and we describe some conventions used in this work. Concepts related to concatenation of words, reachability of states, and partial cover sets are presented.

2.1 Finite State Machines

Let A be an alphabet. Then A^* is the set of all finite sequences of symbols, or words, over A. The length of a word $\rho \in A^*$ will be denoted by $|\rho|$, and ε will denote the empty word. So, $|\varepsilon| = 0$. The concatenation, or juxtaposition, of two words $\alpha, \beta \in A^*$ will be indicated by $\alpha\beta$.

A FSM is a tuple $M = (X, Y, S, s_0, \delta, \lambda)$, where

- X is a finite input alphabet,

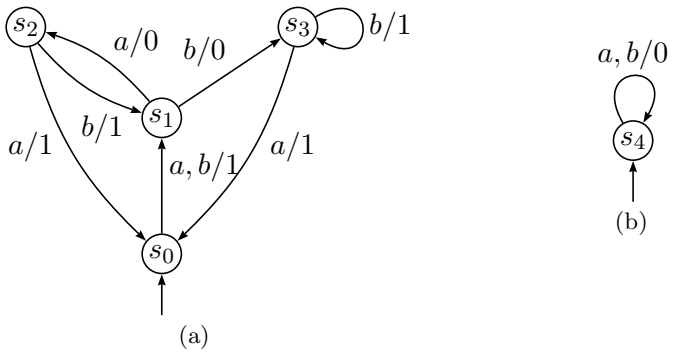

Figure 1: Finite State Machines.

- Y is a finite output alphabet,
- S is the set of states,
- $s_0 \in S$ is the initial state,
- $\delta : X \times S \to S$ is the transition function, and
- $\lambda : X \times S \to Y$ is the output function.

Hereafter, we consider FSMs $M = (X, Y, S, s_0, \delta, \lambda)$ and $M' = (X, Y', S', s'_0, \delta', \lambda')$. Note that they have the same input alphabet, X.

Let $r, s \in S$, $a \in X$ and $b \in Y$ such that $\delta(a, r) = s$ and $\lambda(a, r) = b$. We say that, in state r, and given input a, M changes to state s, and yields b as output. We denote this by $r \xrightarrow{a/b} s$.

Sequential input symbols are represented by a word $\rho \in X^*$, and sequential output symbols are represented by a word $\sigma \in Y^*$. The end state after the successive application of each input is given by the extended function $\widehat{\delta} : X^* \times S \to S$, defined by

$$\begin{aligned} \widehat{\delta}(\varepsilon, s) &= s, \\ \widehat{\delta}(a\rho, s) &= \widehat{\delta}(\rho, \delta(a, s)), \end{aligned}$$

where $a \in X$, $\rho \in X^*$ and $s \in S$.

Similarly, the output extended function is $\widehat{\lambda} : X^* \times S \to Y^*$, defined by

$$\begin{aligned} \widehat{\lambda}(\varepsilon, s) &= \varepsilon, \\ \widehat{\lambda}(a\rho, s) &= \lambda(a, s)\widehat{\lambda}(\rho, \delta(a, s)). \end{aligned}$$

If $\widehat{\delta}(\rho, s) = r$ and $\widehat{\lambda}(\rho, s) = \sigma$, we may write $s \xrightarrow{\rho/\sigma} r$, and, if we are not interested in the output, we may write $s \xrightarrow{\rho} r$.

Usually, a FSM is represented by a state diagram. Figure 1 illustrates two FSMs with initial states s_0 and s_4, respectively. We will refer to this figure through this paper.

2.2 Concatenation of words and relative concatenation

We adopt the usual notation of concatenation of two sets of words, and denote by X_n the set of all input words with length at most n. For the sake of completeness, we give a definition below.

Definition 1. *Let $A, B \subseteq X^*$, $\rho \in X^*$, and let n be a non-negative integer. Then*

- $AB = \{\alpha\beta | \alpha \in A, \beta \in B\}$,
- $\rho A = \{\rho\}A$,
- $A\rho = A\{\rho\}$,
- $X^n = \{\rho \in X^* | \ |\rho| = n\}$, *and*
- $X_n = \bigcup_{k=0}^{n} X^k$. ∎

We note that, if i and j are nonnegative integers, then $X_i X_j = X_{i+j}$.

Many FSM test methods are based on sets like X_n, where n is a parameter defined by the method. Such parameter usually corresponds to the difference between the number of implementation and specification states. The set X_n has a great impact on the test suite size, since, as stated by the next lemma, the size of X_n is exponential.

Lemma 2. *If $|X| \geq 2$, then $|X_n| = \frac{|X|^{n+1}-1}{|X|-1}$.*

Proof. We know that $X_n = \bigcup_{k=0}^{n} X^n$, then, since $X^i \cap X^j = \emptyset$ for all $i \neq j$, $|X_n| = |X^0| + |X^1| + |X^2| + \cdots + |X^n| = 1 + |X| + |X|^2 + \cdots + |X|^n = \frac{|X|^{n+1}-1}{|X|-1}$. ∎

In general, when using a FSM test method, a set of input words must be applied to a set of initial states. In order to reach the desired states, a state cover set is concatenated with the set of input words. If to each initial state, it must be applied a specific set of sequences, then the conventional concatenation operation is not useful. To address this problem, the notion of relative concatenation was introduced in [10]. Before precisely describing this new operation, we need the following definitions, where $\mathcal{P}(A)$ is the power set of a set A.

Definition 3. *Let M be a FSM. A state attribution is a function $\mathcal{B} : S \to \mathcal{P}(X^*)$.* ∎

That is, for each state $s \in S$, we have $\mathcal{B}(s) \subseteq X^*$.

Definition 4. *Let M be a FSM and Π be a partition of S. A class attribution is a function $\mathcal{B} : \Pi \to \mathcal{P}(X^*)$.* ∎

That is, for each class $C \in \Pi$, we have $\mathcal{B}(C) \subseteq X^*$.

A state attribution can be induced by a class attribution in a very intuitive fashion. Let M be a FSM and $\mathcal{B}$ be a class attribution over a partition Π, then the induced state attribution is $\overline{\mathcal{B}}$, defined by $\overline{\mathcal{B}}(s) = \mathcal{B}(C)$, for every $s \in C$ and every $C \in \Pi$.

Now, we can define the relative concatenation.

Definition 5. *Let M be a FSM, $A \subseteq X^*$, and $\mathcal{B}$ a state attribution of M. Given a state s, we define the s-relative concatenation of A and $\mathcal{B}$ as $A \underset{s}{\otimes} \mathcal{B} = \{\alpha\beta | \alpha \in A, \beta \in \mathcal{B}(\widehat{\delta}(\alpha, s))\}$.* ∎

Whenever $s = s_0$, we leave the state index out the operator symbol, and the relative concatenation is written as $A \otimes \mathcal{B}$. If $\mathcal{B}$ is a class attribution, then we may write $A \underset{s}{\otimes} \mathcal{B}$ to mean $A \underset{s}{\otimes} \overline{\mathcal{B}}$.

The usual concatenation may be thought of as a particular case of the relative concatenation, as observed below.

Observation 6. *Let M be a FSM and $A, B \subseteq X^*$. Let also $\mathcal{B}$ be a state attribution such that $\mathcal{B}(s) = B$ for all $s \in S$. Then $A \underset{s}{\otimes} \mathcal{B} = AB$.* ∎

2.3 State Reachability

Now, we define some concepts related to state reachability.

Definition 7. *Let M be a FSM. A state s is reachable if and only if there exists $\rho \in X^*$ such that $s_0 \xrightarrow{\rho} s$. M is connected if and only if every state is reachable.* ∎

Let $C \subseteq S$. We denote by C^r the set of all reachable states of C.

Usually, we are interested in states that are reachable when an input word ρ is applied to a certain state s. We say that a state r is reachable from s using ρ, if $s \xrightarrow{\rho'} r$, where ρ' is a prefix of ρ. In this case, we may omit the input word and say that r is reachable from s. Whenever it is clear from context, we may omit the start state. For example, for machine (a) in Figure 1, s_0, s_1 and s_3 are reachable from s_0 using bb, but only s_3 is reached from s_3 when using bb.

If state r is reachable from state s, then we can ask for a minimum length word ρ such that $s \xrightarrow{\rho} r$. Informally, the length of ρ may be thought of as the distance between s and r. The next observation, although immediate, is a very useful result. We will use this result to bound the distance between s and r by the number of states that ρ can reach.

Observation 8. *Let M be a FSM and let r, s be states of M. Let ρ be a sequence with minimum length such that $r \xrightarrow{\rho} s$. If C is the set of states reached from r using ρ, then $|C| = |\rho| + 1$.* ∎

The output word generated when an input word ρ is applied to a state s depends on the states reachable from s using ρ, with exception of the last one. So, if we consider a set of words of length bounded by a value n, the machine's behavior starting at s will depend on the states within distance of at most n from s. These states around s form a *neighborhood*, that we define as follows.

Definition 9. *Let M be a FSM and k an integer.*

1. *The k-*radius *of a state s, denoted by $\mathrm{rad}(s, k)$, is the set of states that can be reached starting at s applying input words of length at most k. That is, $r \in \mathrm{rad}(s, k)$ if and only if there exists an input word ρ such that $s \xrightarrow{\rho} r$ and $|\rho| \leq k$.*

2. *The* k-neighborhood *of a set of states* C*, denoted by* $\mathrm{nbh}(C,k)$*, is the union of the* k*-radiuses of states in* C*. That is,* $\mathrm{nbh}(C,k) = \bigcup_{s\in C} \mathrm{rad}(s,k)$. ∎

Notice that k may be negative. In this case, the radius of a state and the neighborhood of a set of states are empty.

A neighborhood can be obtained by a recursive algorithm, inspired in the following theorem.

Theorem 10. *Let* M *be a FSM,* C *be a set of states and* s *be a state. If* i, j *are non-negative integers, then*

1. $\mathrm{nbh}(\mathrm{nbh}(C,i),j) = \mathrm{nbh}(C, i+j)$.
2. $\mathrm{nbh}(\mathrm{rad}(s,i),j) = \mathrm{rad}(s, i+j)$.

Proof. We prove the first statement, since the second is a particular case.

First, let $t \in \mathrm{nbh}(\mathrm{nbh}(C,i),j)$. Then there exists $s \in \mathrm{nbh}(C,i)$ such that $t \in \mathrm{rad}(s,j)$, and then there exists $\beta \in X_j$ such that $s \xrightarrow{\beta} t$. Similarly, since $s \in \mathrm{nbh}(C,i)$, there exist $r \in C$ and $\alpha \in X_i$ such that $r \xrightarrow{\alpha} s$. We have $r \xrightarrow{\alpha\beta} t$, $r \in C$ and $\alpha\beta \in X_i X_j = X_{i+j}$, then $t \in \mathrm{nbh}(C, i+j)$.

Now, let $t \in \mathrm{nbh}(C, i+j)$. Then there exist $\gamma \in X_{i+j}$ and $r \in C$ such that $r \xrightarrow{\gamma} t$. Since $X_i X_j = X_{i+j}$, we can obtain $\alpha \in X_i, \beta \in X_j$ such that $\gamma = \alpha\beta$. Let $s \in S$ such that $r \xrightarrow{\alpha} s \xrightarrow{\beta} t$. It follows that $s \in \mathrm{nbh}(C,i)$ and $t \in \mathrm{rad}(s,j)$. Therefore $t \in \mathrm{nbh}(\mathrm{nbh}(C,i),j)$. ∎

For example, from machine (a) in Figure 1, we have

$$\begin{array}{rlll} \mathrm{nbh}(\{s_0\},1) = & \mathrm{rad}(s_0,1) & = \{s_0, s_1\}, & \text{and} \\ & \mathrm{rad}(s_1,1) & = \{s_1, s_2, s_3\}. & \end{array}$$

From the last theorem, we have

$$\begin{array}{rcl} \mathrm{nbh}(\{s_0\},2) & = & \mathrm{nbh}(\mathrm{nbh}(\{s_0\},1),1) = \mathrm{nbh}(\{s_0,s_1\},1) \\ & = & \mathrm{rad}(\{s_0\},1) \cup \mathrm{rad}(\{s_1\},1) \\ & = & \{s_0,s_1\} \cup \{s_1,s_2,s_3\} = \{s_0,s_1,s_2,s_3\}. \end{array}$$

2.4 Cover sets

Cover sets are used in many FSM test methods, in order to guarantee that every state is reached, and that every transition is exercised at least once. But, if we know that some states have already been tested, then we do not need to reach them or exercise its corresponding transitions. In this situation, only untested states must be covered, and so partial cover sets are used. Partial state cover sets and partial transition cover sets are defined next.

Definition 11. *Let* M *be a FSM and* C *be a set of states. A set* $Q \subseteq X^*$ *is a partial state cover set for* C *if, for every state* $s \in C$*, there exists* $\rho \in Q$ *such that* $s_0 \xrightarrow{\rho} s$. ∎

Definition 12. *Let M be a FSM and C be a set of states. A set $P \subseteq X^*$ is a partial transition cover set for C if, for every state $s \in C$ and every symbol $a \in X$, there exist $\rho, \rho a \in P$ such that $s_0 \xrightarrow{\rho} s$.* ∎

Every partial transition cover set P includes a corresponding partial state cover set Q. More precisely, for each partial transition cover set P, one corresponding partial state cover set Q is a subset of P, such that, for every state $s \in C$ and every symbol $a \in X$, there exist $\rho \in Q$ and $\rho, \rho a \in P$ with $s_0 \xrightarrow{\rho} s$. Whenever C is the set of all states, Q and P are, in fact, a state cover set and a transition cover set, respectively, as defined in [10].

A transition cover set may be obtained from a labeled tree for M [4]. A procedure to construct the labeled tree is given in [2]. Although that is intended to cover the entire set of states, one can modify this procedure to obtain a partial cover set, in a straightforward way.

3 State equivalence and state separators

In this section, we define state equivalence. We also introduce the essential notion of a separator, that will be used to compare two given subsets of states.

3.1 State equivalence

Definition 13. *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 $\rho \in X^*$. 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 is ρ-distinguishable from s', and we write $s \not\approx_\rho s'$.*

2. *Let $K \subseteq X^*$. 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 is K-distinguishable from s', 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'. In this case, we write $s \approx_k s'$. Otherwise, s is k-distinguishable from s', and we write $s \not\approx_k s'$.*

4. *State s is equivalent to s' if s is ρ-equivalent to s' for every $\rho \in X^*$. In this case, we write $s \approx s'$. Otherwise, s is distinguishable from s', and we write $s \not\approx s'$.* ∎

As an example, in Figure 1, state s_1 of machine (a) is bb-distinguishable from state s_4 of machine (b), so we write $s_1 \not\approx_{bb} s_4$.

If M and M' are the same machine, the definition above can be taken as specifying equivalence relations over sets of states $C \subseteq S$. In this case, for a set of input words $R \subseteq X^*$, the relation $\approx_R$ induces a partition of the states in C. We denote such partition by $[C/R]$. For example, in Figure 1(a), with $C = \{s_0, s_1, s_2, s_3\}$, $R = \{aaaa\}$ induces the partition $[C/R] = \{\{s_0\}, \{s_1\}, \{s_2, s_3\}\}$.

Next, we make some observations about the partitioning induced by relation $\approx_R$.

Observation 14. *Let M and M' be FSMs and $A, B \subseteq S$, $C \subseteq S'$ and $R \subseteq X^*$. Then*

1. *if for every pair $r \in A, s \in B$ we have $r \not\approx_R s$, then $[(A \cup B)/R] = [A/R] \cup [B/R]$ and $|[(A \cup B)/R]| = |[A/R]| + |[B/R]|$;*
2. *if for every $r \in A$ there exists $s \in B$ such that $r \approx_R s$, then $|[(A \cup B)/R]| = |[B/R]|$;*
3. *if for every $r \in C$ there exists $s \in B$ such that $r \approx_R s$, then $|[C/R]| \leq |[B/R]|$;*
4. *if $A \subseteq B$ then $|[(B \setminus A)/R]| + |[A/R]| \geq |[B/R]|$.* ∎

The number of pairwise distinguishable states of a FSM is called its index, as defined below.

Definition 15. *Let M be a FSM and C be a set of states. The number of equivalence classes induced by the $\approx$ relation over C is denoted by $\iota(C)$. The index of M is $\iota(S)$. If $\iota(S) = |S|$, then the machine is said to be minimal.* ∎

3.2 State separators

From Definition 13, we know that two states s and r are distinguishable if and only if there exists a sequence γ such that $s \not\approx_\gamma r$. Whenever this happens, we say that γ separates s and r. We extend this notion, so that we can *separate* states of two sets. In this case, we use a collection of input sequences instead of just one sequence. This concept is formalized below.

Definition 16. *Let M be a FSM, let A, B be two subsets of states, not necessarily disjoint, and let $R \subseteq X^*$ be a set of input words. R is a (A, B)-separator if and only if for each pair of distinguishable states s and r, such that $s \in A$ and $r \in B$, we have $s \not\approx_R r$.* ∎

To exemplify this new concept, consider machine (a) in Figure 1, and let $A = \{s_0, s_1\}$, $B = \{s_0, s_2\}$ and $C = \{s_0, s_3\}$. The set of input sequences $R = \{ab\}$ is a (A, B)-separator, but, since $s_2 \approx_R s_3$, and $s_2 \in B, s_3 \in C$ are two distinguishable states, R is not a (B, C)-separator. Note that state s_0 is a common element of A and B.

Two special separator cases are:

- A (S, S)-separator is a *characterization set* for M.
- An *identification set* for a state s is any $(\{s\}, S)$-separator.

Notice that, in this paper, we adopt a more flexible definition of characterization sets than that found in [11]. In the latter, the FSM being minimal is a necessary condition for the existence of a characterization set, while in our definition, any FSM has a characterization set. The same happens with respect to identification sets as defined in [10]. We don't even require a characterization set or an identification set to be minimal.

Note that, in Definition 16, sets A and B may have a nonempty intersection. This happens in the case of characterization sets, which are used to separate any pair of distinguishable states of the machine. Actually, there is no restriction to what sets of states we

may select. For instance, we may separate only pairs of states in a given class C with a (C, C)-separator, what we call a *partial characterization set*. On the other hand, if $R \subseteq X^*$, $A, B \subseteq S$ are such that, for every pair $r \in A$, $s \in B$, it is the case that $r \not\approx_R s$, then R is called a *strict* (A, B)-separator.

The concept of separators is a powerful tool that will support test case generation methods. In fact, we can make a separator as strong as we want. This is so because a set of input sequences can assume, simultaneously, different roles for different states: it can be an identification set for s, a partial characterization set for a set of states C, and a strict separator for sets of states A, B. In Section 5, R is a separator that exemplifies this situation.

The next lemma points out some simple observations about separators.

Lemma 17. *Consider a FSM, M. Let A, B, C and D be subsets of states, not necessarily disjoint, and let T and U be sets of input sequences. Let also r and s be states of M. Then,*

1. *T is a (A, B)-separator if and only if T is a (B, A)-separator;*
2. *if T is a (A, B)-separator and U is a (C, D)-separator, then $T \cup U$ is a $(A \cup C, B \cap D)$-separator;*
3. *if T is a strict (A, B)-separator, $r \in A$ and $r \approx_T s$, then $s \notin B$;*
4. *if T is a (A, B)-separator, $r \in A$, $s \in B$ and $r \approx_T s$, then $r \approx s$;*
5. *if T is a (A, B)-separator, $C \subseteq A$ and $D \subseteq B$, then T is a (C, D)-separator.* ∎

We can take one separator and use it as a base to construct another one. With the next lemmas and corollary, we obtain a partial characterization set from a weaker special kind of separator.

Lemma 18. *Let M be a FSM. Let $C \subseteq S$ be a set of states, $B = \mathrm{nbh}(C, 1)$ be its close neighborhood and let T be a $(B, B \backslash C)$-separator such that T partitions C in at least n classes, that is, $|[C/T]| \geq n$. If there exist two distinguishable states $r, s \in C$ such that $r \approx_T s$, then $XT \cup T$ separates C in at least $n+1$ classes, that is, $|[C/(XT \cup T)]| \geq n+1$.*

Proof. For each pair $\hat{r}, \hat{s} \in C$ of states such that $\hat{r} \not\approx \hat{s}$ and $\hat{r} \approx_T \hat{s}$, define $\gamma_{\hat{r}\hat{s}}$ as a sequence with minimum length such that $\hat{r} \not\approx_{\gamma_{\hat{r}\hat{s}}} \hat{s}$. Choose states r and s in C such that γ_{rs} has minimum length. Note that $|\gamma_{rs}| \geq 1$ since $r \not\approx_{\gamma_{rs}} s$. Let $\gamma \in X^*$, $a \in X$ be such that $\gamma_{rs} = a\gamma$. Let r', s' be states such that $r \xrightarrow{a} r'$ and $s \xrightarrow{a} s'$. To prove the lemma, we only need to establish that $r \not\approx_{aT} s$.

First consider the case $|\gamma| = 0$. We then have $\gamma_{rs} = a$, and so $r \not\approx_a s$. Clearly, $r \not\approx_{aT} s$. Now consider the case $|\gamma| \geq 1$. Since γ_{rs} is minimal, we have $\lambda(a, r) = \lambda(a, s)$. It follows that $r' \not\approx_\gamma s'$, otherwise we would obtain the contradiction $\widehat{\lambda}(\gamma_{rs}, r) = \lambda(a, r)\widehat{\lambda}(\gamma, r') = \lambda(a, s)\widehat{\lambda}(\gamma, s') = \widehat{\lambda}(\gamma_{rs}, s)$. If either s' or r' is in $B \setminus C$, then $r' \not\approx_T s'$, because T is a $(B, B \backslash C)$-separator. Again, we get $r \not\approx_{aT} s$. Now, assume that $r', s' \in C$. If $r' \approx_T s'$,

then, since $r' \not\approx_\gamma s'$, the minimal sequence that distinguishes r' and s', $\gamma_{r's'}$, has length $|\gamma_{r's'}| \leq |\gamma| < |\gamma_{rs}|$. This is a contradiction to the choice of r and s. So $r' \not\approx_T s'$, and again $r \not\approx_{aT} s$.

In any case, from $r \not\approx_{aT} s$, we have $r \not\approx_{XT} s$. This means that XT separates two states in the same class of $|[C/T]|$. Hence $|[C/(XT \cup T)]| \geq |[C/T]| + 1$. ∎

Suppose we applied the last lemma and obtained a new separator X_1T. If there exist two distinguishable states in C that are X_1T-equivalent, then we may use the lemma again to obtain a stronger separator, X_2T. In fact, the lemma may be used several times successively. We do this in the following.

Lemma 19. *Let M be a FSM. Let $C \subseteq S$ be a set of states, $B = \mathrm{nbh}(C, 1)$ be its close neighborhood and let T be a $(B, B\backslash C)$-separator such that T partitions C in at least n classes, that is, $|[C/T]| \geq n$. If m is an upper bound on the number of equivalence classes in C, and l is an integer such that $n \leq l \leq m$, then $X_{l-n}T$ separates C in at least l classes, that is, $|[C/X_{l-n}T]| \geq l$.*

Proof. We prove this by induction on l.

For the base, let $l = n$. In this case, the lemma states that $|[C/X_0T]| = |[C/T]| \geq n$, which is true from the hypothesis. Now, consider the case $l > n$ and suppose the lemma is valid for $l - 1$. From the induction hypothesis, $|[C/X_{(l-1)-n}T]| \geq l - 1$. Note that $T \subseteq X_{(l-1)-n}T$ and so $X_{(l-1)-n}T$ is also a $(B, B\backslash C)$-separator. Then, from Lemma 18, we get $|[C/X_{l-n}T)]| = |[C/X_1(X_{(l-1)-n}T)]| \geq (l-1)+1 = l$, as desired. This completes the proof. ∎

Finally, we obtain a partial characterization set for C using the next corollary, which follows directly from Lemma 19.

Corollary 20. *Let M be a FSM. Let $C \subseteq S$ be a set of states, $B = \mathrm{nbh}(C, 1)$ be its close neighborhood and let T be a $(B, B\backslash C)$-separator such that T partitions C in at least n classes, that is, $|[C/T]| \geq n$. If m is an upper bound on the number of equivalence classes in C, then $X_{m-n}T$ is a (C, C)-separator.* ∎

This corollary generalizes a known result from Chow [4], demonstrated in [2], that gives us the ability to generate characterization sets. In fact, we can enunciate the latter a particular case of Corollary 20, when $C = S$, as follows.

Observation 21. *Let M be a FSM with index m. Let $T \subseteq X^*$ such that T partitions the states of M in at least n equivalence classes. Then $X_{m-n}T$ will distinguish every pair of nonequivalent states of M.* ∎

One way of obtaining a separator for two sets of states, A and B, is selecting the minimal subset, R, of a characterization set for the reduced FSM, such that R is a (A, B)-separator. Standard methods to reduce a FSM and to obtain a characterization set for it are known [11]. Although this is a simple procedure that can be used for any FSM, shorter separators may be obtained if we take into consideration the particularities of the FSM being tested.

4 Combined Finite State Machines

Many systems are actually aggregations of other, smaller, subsystems. When modeling such systems, its usual to adopt the *building block strategy* for the development cycle, in which each subsystem is designed, implemented and tested separately. Though each individual part of the system is tested and deemed correct, we have no guarantee that the integrated final implementation is correct.

To use traditional test generation methods for FSMs, such as the W-method [4] and the G-method [2], one should ignore subsystems abstraction and generate test suite for the entire system, as a single block. Since these methods are intended for use against untested implementations, they are inefficient in this situation. In the next section, we will extend these methods to deal with combined FSMs.

The next definition ensures that a state of a subsystem behaviors exactly in the same way, regardless whether it is considered as a state of an isolated submachine, or an incorporated state of the combined machine.

Definition 22. *Let $M = (X, Y, S, s_0, \delta, \lambda)$ be a FSM. A FSM $\dot{N} = (\dot{X}, \dot{Y}, \dot{S}, \dot{s_0}, \dot{\delta}, \dot{\lambda})$ is called a submachine of M if and only if $\dot{X} = X$, $\dot{Y} \subseteq Y$, $\dot{S} \subseteq S$ and, for every $a \in X$ and $s \in \dot{S}$, we have $\dot{\delta}(a, s) = \delta(a, s)$ and $\dot{\lambda}(a, s) = \lambda(a, s)$.* ∎

An easy induction immediately implies the following fact.

Observation 23. *Let $M = (X, Y, S, s_0, \delta, \lambda)$ be a FSM and let $\dot{N} = (\dot{X}, \dot{Y}, \dot{S}, \dot{s_0}, \dot{\delta}, \dot{\lambda})$ be a submachine of M. Let also $s \in \dot{S}$ be as state of M and $\rho \in X^*$, with $s \xrightarrow{\rho} r$ and $r \in S$. Then, in fact, $r \in \dot{S}$.*

A combined FSM is formed by conjoining one or more submachines. This implies that any FSM can be constructed by adding new states and new transitions to connect a set of submachines. Since each subsystem may have only one entry point, every transition that enters a submachine should end in that submachine initial state.

Definition 24. *Let M be a FSM and N be a set of submachines of M. Define $S_N = \{s \in \dot{S} | \dot{N} \in N\}$ the set of all submachine states, and $S_M = S \setminus S_N$ the set of additional states. Also, define $I_N = \{\dot{s_0} | \dot{N} \in N\}$ the set of all submachine initial states. M is N-combined if and only if $s_0 \in S_M$ and, for every pair of states, $s \in S_M$ and $r \in S_N$, if there exists $a \in X$ such that $s \xrightarrow{a} r$, then $r \in I_N$.* ∎

In the sequel, we shall be using the notation introduced in Definitions 22 and 24, that is, given a machine M and a set of submachines N, we have already defined submachines $\dot{N}$ in N, and the sets S_M, S_N, I_N.

In Figure 2, we illustrate a combined FSM. The set of submachines, N, is formed by the machines defined in Figure 1. For this machine, we have $S_N = \{s_0, s_1, s_2, s_3, s_4\}$, $I_N = \{s_0, s_4\}$ and $S_M = \{s_5, s_6\}$. The initial state is $s_5 \in S_M$. We notice that, in fact, this machine satisfies the properties of Definition 24. For example, for states $s_5 \in S_M$ and $s_0 \in S_N$, since $s_5 \xrightarrow{b} s_0$, $s_0 \in I_N$.

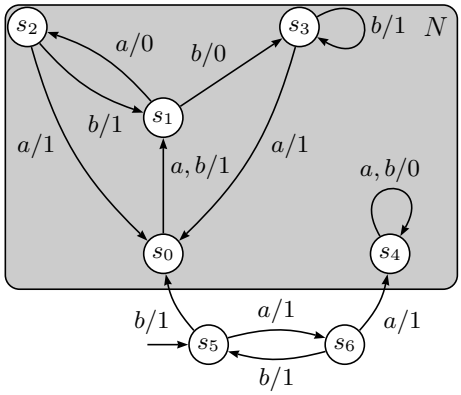

Figure 2: A Combined Finite State Machine.

5 The C-method

We present a new method, named the C-method, to test combined FSM specifications. We assume that an implementation is also a combined FSM in which each submachine is already tested and deemed correct. Also, the number of additional states is limited to a fixed upper bound. If these conditions are satistfied, then the C-method yields a test case suite with full fault coverage for combined FSM specifications.

A submachine can be itself a combined FSM. It can, of course, also be tested using the C-method, giving rise to a recursive testing approach. Notice that the set of submachines may be empty, so one can always use the C-method to directly test FSM specifications. In this particular case, using the C-method is equivalent to using the G-method [2]. Also, notice that it is necessary to test a submachine only once, and then implementations that use it can be tested several times at a minimum cost. Further, incremental testing is possible, so that, if the specification is changed, only the affected submachines need to be retested.

Next, we formalize our fault model. Then, we describe the construction of the test suite.

5.1 The fault model

The system specification M is a combined FSM, obtained from a set N of submachines. We assume that

1. M is connected, and
2. for every pair of states, $s \in S_M$ and $r \in S_N$, we have $s \not\approx r$.

These assumptions are reasonable, since there is no meaning in having unreachable states in the specification, or in reimplementing the behavior of an already available submachine state.

We presume that each submachine $\dot{N} \in N$ has a correct implementation $\dot{N}'$, and denote the set of submachine implementations by N'. One system implementation M' is obtained by combining the set of submachines N', using up to m additional states.

The fault model is formed by (N', m)-combined FSM cadidate implementations, as defined below.

Definition 25. *Let M be a FSM specification and let N be a set of submachines of M such that M is N-combined. Let N' be a set of FSMs and m be a positive integer. A FSM candidate implementation M' is (N', m)-combined if*

1. *M' is N'-combined;*
2. *$\iota(S_M) \leq |S'_M| \leq m$;*
3. *For every $\dot{N} \in N$, there exists $\dot{N}' \in N'$ such that $\dot{s_0} \approx \dot{s_0}'$;*
4. *For every $\dot{N}' \in N'$, there exists $\dot{N} \in N$ such that $\dot{s_0} \approx \dot{s_0}'$.* ∎

For the ease of notation, we shall write S'_M, S'_N and I'_N, instead of $S_{M'}$, $S_{N'}$ and $I_{N'}$, respectively, for the occasions where M' and N' appear as indexes.

In the following observation, the first two claims may be obtained by an easy induction using hypotheses (3) and (4) of Definition 25, respectively. The last two claims are particular cases.

Observation 26. *Let M be N-combined specification and M' be a (N', m)-combined candidate implementation. Then,*

1. *if $t \in I_N$, $s \in S_N$, and $\rho \in X^*$, such that $t \xrightarrow{\rho} s$, then there exist $t' \in I'_N$ and $s' \in S'_N$, such that $t' \xrightarrow{\rho} s'$, $t \approx t'$ and $s \approx s'$.*
2. *if $t' \in I'_N$, $s' \in S'_N$, and $\rho \in X^*$, such that $t' \xrightarrow{\rho} s'$, then there exist $t \in I_N$ and $s \in S_N$, such that $t \xrightarrow{\rho} s$, $t' \approx t$ and $s' \approx s$.*
3. *if $s \in S^r_N$ ($s \in I_N$), then there exists $s' \in S'_N$ ($s' \in I'_N$) such that $s \approx s'$.*
4. *if $s' \in S'^r_N$ ($s' \in I'_N$), then there exists $s \in S_N$ ($s' \in I_N$) $s' \approx s$.* ∎

The Figure 3 illustrates a candidate implementation for the combined machine depicted in Figure 2. We claim that this combined machine obeys Definition 25 for $m = 4$. Observe that $2 = \iota(S_M) \leq |S'_M| \leq m = 4$. We check that for each state in S_N, there exists a corresponding state in S'_N. For instance, we have $s_0 \in S_N$ and $r_4 \in S'_N$ such that $s_0 \approx r_4$.

Though we assume that the candidate implementation is (N', m)-combined, no other information about it is available in general. Therefore, we must treat M' as a *black-box*. Some implications are that it may have unreachable states, such as the state r_{10}, and, in implementations, additional states may be equivalent to submachine states, such as the states r_3 and r_9, respectively.

Also, a submachine implementation must not be minimal. In fact, the number of states used for submachine implementation is usually greater than the number of states of the corresponding submachine specification. For example, r_9 and r_{10} are equivalent states that correspond to state s_4.

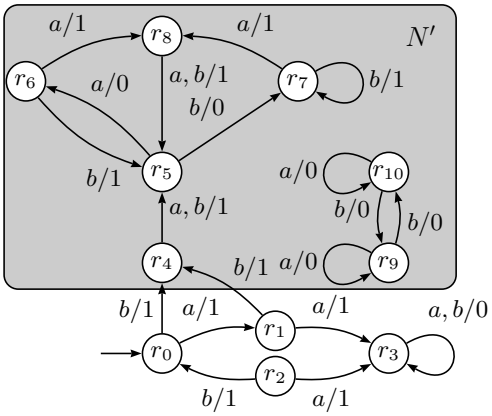

Figure 3: A candidate implementation.

Algorithm 1: Test suite construction for C-method

Input: M, m
Output: π
begin
  Obtain a partial transition cover set P for S_M such that $\varepsilon \in P$;
  Obtain a $(S_M \cup I_N, S_N)$-separator R ;
  Define $l \longleftarrow |[S/R]| - |[S_N/R]|$;
  Choose $l' \leq |[S'/R]| - |[S'_N/R]|$;
  Define $n \longleftarrow \max\{l', l\}$;
  if $m < n$ **then**
    mesg "No correct candidate implementation is possible" ;
  else
    Define $A \leftarrow \text{nbh}(I_N, m-n-1)$;
    Obtain a (A, S_N)-separator T ;
    Let $\mathcal{R}(S_M) \longleftarrow R$ and $\mathcal{R}(S_N) \longleftarrow R \cup T$;
    forall $s \in S$ **do**
      Define $\mathcal{Z}(s) \longleftarrow X_{m-n} \underset{s}{\otimes} \mathcal{R}$;
    Define $\pi \longleftarrow P \otimes \mathcal{Z}$;
    return π
end

5.2 Test suite generation

The C-method is depicted as Algorithm 1. We now expand on each step.

THE INPUT**:** We start with a specification M that is a N-combined FSM, where N is a set of submachines of M. We adopt the notations exposed in Definition 24. The input parameter m is the maximum number of additional states and is determined by the user. We assert that m must be at least as large as the number of equivalence classes of S_M, *i.e.*, $m \geq \iota(S_M)$, otherwise no correct candidate implementation would be possible.

THE COVER SET P**:** P is a partial transition cover set for S_M such that $\varepsilon \in P$. This set is used to reach every additional state in the specification, so that one can exercise the corresponding transitions. Since states in S_N are known to be already correctly implemented, there is no need to cover them. We require $\varepsilon \in P$ for reasons that will be clear in Section 7.

THE SEPARATOR R**:** We select R as any $(I_N \cup S_M, S_N)$-separator. This set takes on several roles, depending on the states we are considering. For example, if we consider R as a strict (S_M, S_N)-separator, we can use R to distinguish additional states from submachine states. As a (I_N, S_N)-separator, R can be used to uniquely identify initial states of submachines, and so on.

THE PARAMETER n**:** The relation $\approx_R$ partitions the states of M. Based on this, we let $l = |[S/R]| - |[S_N/R]|$. Similarly, the relation $\approx_R$ partitions the states of M'. We now *choose* a value l' such that $l' \leq |[S'/R]| - |[S'_N/R]|$. If no information about M' is available, we can always choose $l' = 0$. We then set $n = \max\{l, l'\}$. The larger n is, the less we need to strengthen R in order to obtain a partial characterization set for S'_M. We use G-method [3] ideas, so that, if knowledge is available about the implementation, l' may be set larger than l, thus giving rise to more succinct test suites. We notice that we will always have $m \geq n$, otherwise the input m is not valid. In this case, the algorithm halts and throws a message to the user.

THE SEPARATOR T**:** This is a separator used to complement R, whenever there is a need to uniquely identify a state in a close neighborhood of I_N. We define $A = \text{nbh}(I_N, m-n-1)$ and select T to be any (A, S_N)-separator. Notice that, in the case $m = n$, A contains no element, so we may select T as the empty set.

THE STATE ATTRIBUTION $\mathcal{Z}$**:** We need to use T only for input words that reach states in S_N. Then, to avoid generating unnecessary test sequences, we make use of a class attribution $\mathcal{R}$, defined by $\mathcal{R}(S_N) = T \cup R$ and $\mathcal{R}(S_M) = R$. We define a state attribution $\mathcal{Z}$ such that for each $s \in S$, $\mathcal{Z}(s) = X_{m-n} \underset{s}{\otimes} \mathcal{R}$.

THE TEST SUITE π**:** The test suite generated by C-method is given by $\pi = P \otimes \mathcal{Z}$.

Clearly, $|[S/R]| - |[S_N/R]|$ counts only the classes that are entirely contained in S_M. Hence, the following observation is immediate.

Observation 27. $l \leq |[S_M / R]|$. ∎

6 Comparison with the W-method

We compare our method with the W-method. First, we briefly review the W-method. Then, we give a lower bound to the ratio between the number of test cases generated by W-method and the number of test cases generated by C-method. Next, to exemplify our method, we generate test suites for specification in Figure 2 using W-method and C-method.

6.1 The W-method

The W-method [4] applies to minimal, completely specified and deterministic FSMs. So, to use W-method to test a specification that is not minimal, one must first reduce it. The set of implementation candidates comprehends all faulty machines with up to m_W states. Test suites for this fault model are called m_W-complete.

The method depends on two sets, P_W and W. First, obtain P_W, a transition cover set for S, and W, a characterization set of S. Then, let $Z_W = X_{m_W - n_W} W$, where n_W is the number of specification states. The generated test suite is $\pi_W = P_W Z_W$.

6.2 Comparison analysis

Usually, the efficiency of a test suite is defined as the sum of the lengths of all prefix-free test sequences [3]. A test suite highly depends on the chosen sets that the method uses, that is, the cover sets, separators and characterization sets. Therefore, unless we restrict the analysis to a specific family of FSMs, and express the used sets, it is not possible to calculate the efficiency or even count the number of distinct sequences of a test suite.

We will make a simplified analysis based on the number of test cases directly generated by the method, that is, we assume that a test suite is a multiset, and so may contain prefix or elements with multiplicity greater than one. This is reasonable since we will make a relative comparison between W-method and C-method, and, as we will show, the test suite generated by our method can always be chosen to be a subset of that generated by the W-method. Further, the W-method may generate test sequences which are lengthier then any sequence generated by C-method.

Depending on the procedure used to construct a partial transition cover set P, different sets are obtained. In fact, if P is not minimal, it can be arbitrarily large. We assume that the cover set P_W used by the W-method is obtained from a labeled tree, using the procedure defined in [3]. A cover set P obtained in this way is minimal and have the property that, for every $s \in S$, there exists $\rho_s \in X^*$ such that $s_0 \xrightarrow{\rho_s} s$, and, for every $a \in X$ we have $\rho_s a \in P$.

Lemma 28. *Let M be a FSM, let C be a set of states and P be a partial transition cover set for C. Let $H(s) = \{\rho_s a | a \in X\}$ for each $s \in C$. If $s, r \in C$ and $s \neq r$, then $|H(s)| = |H(r)| = |X|$ and $H(s) \cap H(r) = \emptyset$.*

Proof. Let $r, s \in C$ such that $r \neq s$. Suppose that there exists $\gamma \in P$ such that $\gamma \in H(s) \cap H(r)$. Then, from $\gamma \in H(s)$, $\gamma = \rho_s a$, where $a \in X$. Similarly, $\gamma = \rho_r b$, where

$b \in X$. So $\rho_s a = \rho_r b$, then $a = b$ and $\rho_s = \rho_r$. Now, we have $s = \delta(\rho_s, s_0) = \delta(\rho_r, s_0) = r$. This is a contradiction, and so we may conclude that $H(s) \cap H(r) = \emptyset$. From the definition, $|H(s)| = |H(r)| = |X|$. ∎

We shall use the notation of the proof above in the next two lemmas. The next lemma gives lower and upper bounds on the sizes of transition cover sets obtained from a labeled tree.

Lemma 29. *Let M be a FSM, C be a set of states and P be a partial transition cover set for C obtained from a labeled tree . Then $|X||C| \leq |P| \leq (|X| + 1)|C|$.*

Proof. Let $H(s)$ be as in Lemma 28. Define $Q = \bigcup_{s \in C} \{\rho_s\}$ and $H = \bigcup_{s \in C} H(s)$. From Definition 12, the union $Q \cup H$ is a partial transition cover set for C. Since $Q \cup H \subseteq P$ and P is a minimal partial transition cover set for C, the equality $Q \cup H = P$ must hold. So $|H| \leq |P| \leq |Q| + |H|$. But $|Q| = |C|$ and, from Lemma 28, $|H| = \sum_{s \in C} |H(s)| = |C||X|$. Therefore $|X||C| \leq |P| \leq (|X| + 1)|C|$. ∎

The C-method requires a partial transition cover set, P, for the set S_M. We obtain P by selecting the minimal subset of P_W that is also a partial transition cover set for S_M. The next lemma estimates the proportion of P included in P_W.

Lemma 30. *Let M be a combined FSM. Let P_W be a transition cover set for S obtained from a labeled tree and P be a minimal partial cover set for S_M such that $P \subseteq P_W$. Then $\frac{|P_W|}{|P|} \geq 1 + \frac{|X|}{|X|+1} \frac{|S_N|}{|S_M|}$.*

Proof. Let $H(s) = \{\rho_s a | a \in X\}$, for each $s \in S$. Define $H_N = \bigcup_{s \in S_N} H(s)$. Let $\gamma \in H_N$. Then, there exists $s \in S_N$ such that $\gamma \in H(s)$. So $\gamma = \rho_s a$, where $a \in X$. Since $s \in S_N$, then, from Observation 23, $\delta(a, s) \in S_N$, and so $\delta(a, s) \notin S_M$. Thus, $\gamma \notin P$, otherwise P would not be minimal. It follows that $H \cap P = \emptyset$. Since $H \subseteq P_W$, $|P_W \setminus P| \geq |H|$. Then $\frac{|P_W|}{|P|} = \frac{|P| + |P_W \setminus P|}{|P|} = 1 + \frac{|P_W \setminus P|}{|P|} \geq 1 + \frac{|H|}{|P|}$. Applying Lemmas 28 and 29, we obtain $\frac{|P_W|}{|P|} \geq 1 + \frac{|X|}{|X|+1} \frac{|S_N|}{|S_M|}$. ∎

The test suite generated using the C-method can be constructed as a subset of the test suite generated using W-method . In order to do this, we can use the partial cover set mentioned above, and choose R and T to be the characterization set W itself. But, of course, if one uses weaker separators R and T, instead of W, the gain established by the next theorem can be much more significant. That is, the ratio obtained below may be multiplied by a factor that depends on each FSM, and thus can be much better. We write $\|\mathcal{A}\|$ to denote the cardinality of a multiset $\mathcal{A}$.

Theorem 31. *Let M be a minimal connected FSM and let N a set of submachines of M such that M is N-combined. Assume that $|X| \geq 2$. Consider the fault model defined by all FSM M' such that M' is a (N', m)-combined candidate implementation for M, $|S'_M| = m$, and $|S'_N| = k$. If π_W is a complete test suite generated using the W-method, then we can obtain a complete test suite π using the C-method in such a way that*

1. *$|\pi| \in O(l(j + l)^2 |X|^{m-l+1})$,*

2. $|\pi_W| \in O((j+l)^3|X|^{m-l+k-j+1})$, *and*

3. $\frac{\|\pi_W\|}{\|\pi\|} \geq \left(1 + \frac{|X|}{|X|+1}\frac{j}{l}\right)\left(|X|^{k-j}\right)$,

where $l = |S_M|$ *and* $j = |S_N|$.

Proof. We first inspect the W-method. Note that a candidate implementation has $|S'| = |S'_M| + |S'_N| = m + k$ states and that the specification has $|S| = |S_M| + |S_N| = j + l$ states. Then, we must choose parameter $m_W \geq m + k$, otherwise the candidate implementation would not fit the fault model of the W-method. Also, $n_W = j + l$. Let W be a minimal characterization set for M and P_W be a transition cover set for S obtained from a labeled tree. Then $\pi_W = P_W X_{m_W - n_W} W \subseteq P_W X_{(k+m)-(j+l)} W$.

Now, we examine the C-method.

Cover set P: Define P as the minimal partial transition cover set for S_M with $P \subseteq P_W$.

Separator R and T: Since W is a (S, S)-separator, from Lemma 17(5), W is a (S_M, S_N)-separator and a (A, S_N)-separator. Select $R = W$ and $T = W$.

Parameter n: No information about the implementation is available, so we choose $l' = 0$. Then, $n = max\{l', l\} = l$. Since R is a characterization set and M is minimal, we have $l = |[S/R]| - |[S_N/R]| = |S| - |S_N| = |S_M|$.

State attribution $\mathcal{Z}$: We set $\mathcal{R}(S_N) = T \cup R = W$ and $\mathcal{R}(S_M) = R = W$. Since $\mathcal{R}(S_M) = \mathcal{R}(S_M) = W$, from Observation 6, $\mathcal{Z}(s) = X_{m-n} \underset{s}{\otimes} \mathcal{R} = X_{m-n}W = X_{m-l}W$, for every $s \in S$.

Test suite π: From Observation 6 again, $\pi = P \otimes \mathcal{Z} = PX_{m-l}W$.

We obtain the first item, the second is analogous. Notice that, since W is minimal, it contains at most one element for each pair of states of S, so clearly $|W| \leq |S|^2$. We have

$$\begin{aligned}|\pi| &= |PX_{m-l}W| \leq |P||X_{m-l}||W| \\ &\leq (|X|+1)|S_M|\frac{(|X|^{m-l+1}-1)}{(|X|-1)}|S|^2 \in O(l(j+l)^2|X|^{m-l+1}). \quad \text{(Lemmas 2 and 29)}\end{aligned}$$

Finally, we calculate the desired ratio

$$\begin{aligned}\frac{\|\pi_W\|}{\|\pi\|} &\geq \frac{\|P_W X_{(k+m)-(j+l)} W\|}{\|PX_{m-l}W\|} \\ &= \frac{|P_W|}{|P|}\frac{|X_{(k+m)-(j+l)}|}{|X_{m-l}|}\frac{|W|}{|W|} \\ &= \frac{|P_W|}{|P|}\frac{(|X|^{(k+m)-(j+l)+1}-1)/(|X|-1)}{(|X|^{m-l+1}-1)/(|X|-1)} \quad \text{(Lemma 2)} \\ &= \frac{|P_W|}{|P|}|X|^{k-j}\frac{\left(|X|^{m-l+1} - \frac{1}{|X|^{k-j}}\right)}{(|X|^{m-l+1}-1)} \geq \frac{|P_W|}{|P|}|X|^{k-j}\frac{(|X|^{m-l+1}-1)}{(|X|^{m-l+1}-1)} \\ &= \frac{|P_W|}{|P|}|X|^{k-j} \geq \left(1 + \frac{|X|}{|X|+1}\frac{j}{l}\right)|X|^{k-j}. \quad \text{(Lemma 30)}\end{aligned}$$

∎

Each factor of the ratio given by Theorem 31 has a special meaning. The first factor is proportional to $\frac{j}{l}$, that is, the greater the number of submachines states, j, when compared to the number of additional states, l, the greater the advantage of using the C-method. This is expected, since we do not need to test any of the submachine states.

Before addressing the second factor, we observe that an implementation needs not to be minimal. In fact, the number of states in an implementation is usually greater than the number of states in a specification. The difference between these numbers correspond to what we call *superfluous* states. As noted in Section 2, the size of complete test suites for FSMs is usually exponential on the number of superfluous states. However, in the C-method, no superfluous state in any of the submachines is considered. The implication of this is that the C-method may generate test suites with exponentially less test cases than the W-method. A measure of this gain is given by the second factor, where the exponent is $k - j$, the difference between the number of implementation submachine states and the number of specification submachine states.

The C-method is scalable. That is, unlike the W-method and the G-method, which require that specifications have a small number of states, using the C-method, we can test systems with a high number of states, provided that the number of additional states is kept low. This is due the fact that, in spite of the specification being arbitrarily large, we know, observing the first item of the theorem, that the size of the generated test suite is only polynomial on the number of submachines states. Compare this to the bound obtained for W-method. This is a major advantage of using the building block strategy when testing systems.

6.3 Example

We will generate test suites for the specification as depicted in Figure 2. The test suites are intended for (N', m)-combined candidate implementations, with $m = 4$ and where N' is illustrated in Figure 3.

Using the W-method

The specification in Figure 2 has $n_W = 7$ states and the candidate implementation has $|S'_N| = 7$ submachines states and up to $m = 4$ additional states. So the minimum value for m_W we may choose is $m_W = 7 + 4 = 11$. Next, we select a transition cover set, P_W, using a labeled tree [3], then we choose a minimal characterization set, W, and finally the set Z_W is computed.

- $P_W = \{\varepsilon, a, b, aa, ab, aaa, aab, ba, bb, baa, bab, baaa, baab, baba, babb\}$;
- $W = \{aaaa, bb\}$;
- $Z_W = X_{m_W - n_W} W = X_4 W$.

The test suite is $\pi_W = P_W Z_W$ with $|\pi_W| \leq |P_W||X_4||W| = 15 \times 31 \times 2 = 930$ test cases. In fact, π_W has 256 prefix-free words.

Using the C-method

We select P as the minimal subset of P_W that is a partial transition cover set for S_M and then a $(S_M \cup I_N, S_N)$-separator R is obtained from W. Notice that R is a weaker separator than W, since, for example, $s_2 \approx_R s_3$, but $s_2 \not\approx_W s_3$. Next, we first partition the states of the specification and obtain the value l, and then, since no specific information is available about implementation, we choose $l' = 0$. From those two values we obtain the parameter n. Proceeding, we define A as the $(m{-}n{-}1)$-neighborhood of I_N, and then we select a (A, S_N)-separator T from W. Finally, we calculate the state attribution $\mathcal{Z}$:

- $P = \{\varepsilon, a, b, aa, ab, aaa, aab, ba, bb\}$;
- $R = \{aaaa\}$;
- $[S/R] = \{\{s_0\}, \{s_1\}, \{s_2, s_3\}, \{s_4\}, \{s_5\}, \{s_6\}\}$;
- $[S_N/R] = \{\{s_0\}, \{s_1\}, \{s_2, s_3\}, \{s_4\}\}$;
- $l' = 0$, $l = |[S/R]| - |[S_N/R]| = 2$ and so $n = \max\{l', l\} = 2$;
- $A = \text{nbh}(I_N, m{-}n{-}1) = \text{nbh}(\{s_0, s_4\}, 1) = \{s_0, s_1, s_4\}$;
- $T = \{aaaa\}$;
- $\mathcal{R}(S_N) = T \cup R = R$ and $\mathcal{R}(S_M) = R$;
- $\mathcal{Z}(s) = X_{m-n} \underset{s}{\otimes} \mathcal{R} = X_{m-n}R = X_2R$ for every $s \in S$.

The test suite is $\pi = P \otimes \mathcal{Z} = PX_2R$ with $|\pi| \leq |P||X_2||R| = 9 \times 7 \times 1 = 63$ test cases. In fact, π has 20 prefix-free words.

7 Ensuring implementation equivalence

In this section, we prove that the C-method generates a complete test suite.

Throughout this section, let the specification M be a N-combined FSM, where N is a set of submachines of M. We assume that M is connected, and that for every pair of states, $s \in S_M, r \in S_N$, we have $s \not\approx r$. Also, let M' be a (N', m)-combined candidate implementation, where N' is a set of submachines of M'.

We generate a test suite π using Algorithm 1, on page 15, providing M and m as input. In the following, we refer to P, R, l', l, n, A, T, $\mathcal{R}$ and $\mathcal{Z}$ as defined at that algorithm. We denote by Q the minimal partial state cover set associated to P with $\varepsilon \in Q$. Also, we let $Z = X_{m-n}R$, and $A' = \text{nbh}(I'_N, m{-}n{-}1)$.

The proof is divided in three parts. First, we partition the states of S'_M to obtain some auxiliary results. Then, we use the concept of separators to show that Z is a partial characterization set of S'^r_M, so that we can distinguish states of the implementation. Finally, we show that π is a complete test suite.

7.1 Preliminary results

First, we need to ensure that the Algorithm 1 always returns a test suite π. It is enough to verify the assertion $m \geq n$. The next lemma shows that $m \geq n$ holds, unless the user has provided a value of m too small. In this case, no correct candidate implementation is possible, and the algorithm halts yielding a message.

Lemma 32. *If $m < n$, then $m < \iota(S_M)$, and so $s_0 \not\approx s'_0$.*

Proof. Suppose, for contradiction, that $n = l'$. Then,

$$\begin{aligned} m < n = l' &\leq |[S'/R]| - |[S'_N/R]| && (l' \text{ definition}) \\ &\leq |[S'_M/R]| + |[S'_N/R]| - |[S'_N/R]| && (\text{Obs. } 14(4)) \\ &= |[S'_M/R]| \leq |S'_M| \leq m. \end{aligned}$$

This implies $n = l$.

Therefore, from Observation 27,

$$m < n = l \leq |[S_M/R]| \leq \iota(S_M).$$

Now, suppose for contradiction that $s_0 \approx s'_0$. Then, clearly $|[S/R]| = |[S'^r/R]|$. It follows

$$\begin{aligned} |[S_M/R]| = |[S/R]| - |[S_N/R]| &= |[S'^r/R]| - |[S_N/R]| && (1) \\ &\leq |[S'^r_M/R]| + |[S'^r_N/R]| - |[S_N/R]| && (2) \\ &= |[S'^r_M/R]| + |[S_N/R]| - |[S_N/R]| && (3) \\ &= |[S'^r_M/R]| \leq |S'_M| \leq m. \end{aligned}$$

Detailed justifications for each step above are as follows:

1. Since R is a strict (S_M, S_N)-separator, we may use Observation 14(1) to obtain $|[S/R]| = |[S_M/R]| + |[S_N/R]|$.
2. We may simply use Observation 14(4), and get $|[S'^r/R]| \leq |[S'^r_M/R]| + |[S'^r_N/R]|$.
3. From Observation 26(3,4), and since $S_N = S^r_N$, we get $|[S'^r_N/R]| = |[S_N/R]|$.

From the contradiction $m < |[S_M/R]| \leq m$, we have $s_0 \not\approx s'_0$. ∎

In the following simple observation, we show the hierarchy between the sets involved in the test suite π. We use this to obtain needed equivalence relations, when we have a higher equivalence relation in the hierarchy. For example, if we have $s_0 \approx_\pi s'_0$, then we also get $s_0 \approx_{PZ} s'_0$. Since this observation can be easily derived from the Algorithm 1, we may use it without mention.

Observation 33. *It holds:*

1. *$QZ \subseteq PZ \subseteq P \otimes \mathcal{Z} = \pi$, and*

2. for all $s \in S$, $R \subseteq Z \subseteq \mathcal{Z}(s)$.

The following is an auxiliary lemma used to decide whether a reachable implementation state is an additional state, or a submachine state.

Lemma 34. *Let $s \in S_M$, $s' \in S'^r$ be such that $s \approx_R s'$. Then $s' \in S'_M$.*

Proof. Suppose, for the sake of contradiction, that $s' \in S'_N$. Then $s' \in S'^r_N$, and, by Observation 26(4), there exists $r \in S_N$ such that $r \approx s'$. In particular, $r \approx_R s'$, and then $s \approx_R r$. Since R is a strict (S_M, S_N)-separator and $r \in S_N$, from Lemma 17(3) we get $s \notin S_M$. This contradicts the hypothesis, so $s' \in S'_M$. ∎

We know that T is as a (S_N, A)-separator and R is a (S_N, I_N)-separator. The next lemma states that they are also separators for the reachable states of implementation analogous sets I'_N, S'_N and A'.

Lemma 35. *Let S'^{ri}_N be the set of submachine states reachable from I'_N, that is, $s' \in S'^{ri}_N$ if and only if there exist $\rho \in X^*$, $t' \in I'_N$ such that $t' \xrightarrow{\rho} s'$. Then,*

1. T is a (S'^{ri}_N, A')-separator,

2. R is a (S'^{ri}_N, I'_N)-separator,

3. T is a (S'^{r}_N, A')-separator, and

4. R is a (S'^{r}_N, I'_N)-separator.

Proof. For the first claim, if R were not a (S'^{ri}_N, I'_N)-separator, then we would get distinguishable states $s' \in S'^{ri}_N$ and $r' \in I'_N$ with $s' \approx_R r'$. But, by Observation 26(2), we then get $s \in S_N$ with $s \approx s'$, and, by Observation 26(4), we also get $r \in I_N$ with $r \approx r'$. This gives $s \approx_R r$, and since R is a (I_N, S_N)-separator, $s \approx r$, but then, $s' \approx r'$, that is a contradiction.

Similarly, for the second claim, if T were not a (S'^{ri}_N, A')-separator, we would get distinguishable states $s' \in S'^{ri}_N$ and $r' \in A'$ with $s' \approx_T r'$. Using Observation 26(2), we get $s \in S_N$ with $s' \approx s$. Now, note that $A' = \mathrm{nbh}(I'_N, m{-}n{-}1)$. By Definition 9, there is some $t' \in I'_N$ and some $\rho \in X^*$, with $|\rho| \leq m - n - 1$, such that $t' \xrightarrow{\rho} r'$. Now, using Observation 26(2), we obtain $t \in I_N$ and $r \in S_N$, such that $t \xrightarrow{\rho} r$ and $r' \approx r$. It follows that $s \approx_T r$, $s \in S_N$ and $r \in \mathrm{nbh}(I_N, m{-}n{-}1) = A$. Since T is a (S_N, A)-separator, we obtain $s \approx r$, and then the contradiction $s' \approx r'$.

Clearly, we have $S'^{r}_N \subseteq S'^{ri}_N$. Then, the other claims are obtained using Lemma 17(5) and the first two claims. ∎

Consider the four subsets $I, J, K, L \subseteq S'_M$, defined by cases as follows.

Definition 36. *Take $s' \in S'_M$.*

1. If there is $s \in S_M$ such that $s' \approx_R s$. Then:

(a) If there is $\rho \in Q$ with $s_0 \xrightarrow{\rho} s$, $s'_0 \xrightarrow{\rho} s'$ and $s' \approx_Z s$. Then, put s' in I.

(b) Else, put s' in J.

2. *If there is $s \in S_N$ such that $s' \approx_R s$. Then, put s' in K.*

3. *Else,* i.e., *$s' \not\approx_R s$ for all $s \in S$. Then, put s' in L.*

Clearly, L is disjoint from the other three sets. Also, since R is a strict (S_M, S_N)-separator, we know that K is disjoint from $I \cup J$. Obviously I is disjoint from J. Moreover, the union $I \cup J \cup K \cup L$ is clearly S'_M. We collect these simple facts in the following.

Observation 37. *For the four sets $I, J, K, L \subseteq S'_M$ we have:*

1. *They are two by two disjoint.*

2. *$I \cup J \cup K \cup L = S'_M$.* ∎

Since $\varepsilon \in Q$, the following is immediate.

Observation 38. *If $s_0 \approx_\pi s'_0$, then $s'_0 \in I$.*

We will use these sets in combination with Observation 8 to put a bound on the length of certain distinguishing sequences needed in Lemmas 40 and 41. The next lemma estimates the size of each set.

Lemma 39. *If $s_0 \approx_\pi s'_0$, then the following inequalities hold:*

1. *$|I| \geq l$;*

2. *$|L| \geq n - l$;*

3. *$|J| + |K| \leq m - n$;*

4. *$|[S'_M / R]| \geq n$.*

Proof. We will prove each statement in turn.

Claim 1: $|I| \geq l$.

Let $s \in S_M$. Since Q covers all states of S_M, there is $\rho \in Q$ such that $s_0 \xrightarrow{\rho} s$. Take $s' \in S'$ such that $s_0 \xrightarrow{\rho} s'$. From $s_0 \approx_\pi s'_0$, we obtain $s_0 \approx_{QZ} s'_0$, and then, from $\rho \in Q$, we get $s_0 \approx_{\rho Z} s'_0$. It follows that $s \approx_Z s'$, and, in particular, $s \approx_R s'$. Using Lemma 34, we get $s' \in S'_M$, and then, clearly, $s' \in I$.

We have shown that for each $s \in S_M$ there is a corresponding state $s' \in I$ such that $s \approx_R s'$. Using Observations 27 and 14(3), we have $l \leq |[S_M / R]| \leq |[I / R]| \leq |I|$.

Claim 2: $|L| \geq n - l$.

If $n = l$, then clearly the claim holds. Let now $n = l'$. We have the following.

$$\begin{aligned}
l &= |[S/R]| - |[S_N/R]| = |[S_N \cup S_M/R]| - |[S_N/R]| && (1)\\
&= |[S_N/R]| + |[S_M/R]| - |[S_N/R]| = |[S_M/R]|, && (2)\\
|L| &\geq |[L/R]| = |[L/R]| + |[S_M/R]| - l && (3)\\
&\geq |[L/R]| + |[I \cup J/R]| - l && (4)\\
&= |[I \cup J \cup L/R]| - l && (5)\\
&= |[S'_M \setminus K/R]| - l && (6)\\
&\geq |[S'/R]| - |[S'_N \cup K/R]| - l && (7)\\
&= |[S'/R]| - |[S'_N/R]| - l && (8)\\
&= l' - l = n - l. && (9)
\end{aligned}$$

Detailed justifications for each step above are as follows:

1. From l definition and, from Definition 24, $S = S_M \cup S_N$.
2. Because R is a strict (S_M, S_N)-separator, and using Observation 14(1).
3. The inequality is obvious. For the equality, note that item (2) implies $l = |[S_M/R]|$.
4. From Observation 14(3) and Definition 36(1), we immediately get $|[I \cup J/R]| \leq |[S_M/R]|$.
5. Let $r' \in I \cup J$ and $s' \in L$. From Definition 36(1), there exists $t \in S_M$ such that $r' \approx_R t$. Suppose that $s' \approx_R r'$, then $s' \approx_R t$. But, since $t \in S$, from Definition 36(3), $s' \notin L$. This is a contradiction, so $r' \not\approx_R s'$. Now, we may use Observation 14(3).
6. Immediate from Observation 37(1,2).
7. Immediate from Observation 14(4).
8. Let $r' \in K$. Then, from Definition 36(3), there exist $s \in S_N$ such that $r' \approx_R s$. From Observation 26(3) and since $S_N = S_N^r$, there exists $s' \in S'_N$, such that $s \approx_R s'$, and so $r' \approx_R s'$. Now, we may use Observation 14(2).
9. This is from l' definition.

Claim 3: $|J| + |K| \leq m - n$.

We have $|J| + |K| = |S'_M| - |I| - |L|$, by Observation 37(1,2). Since $m \geq |S'_M|$, we can use the previous two claims, and state $|J| + |K| \leq m - l - (n - l) = m - n$.

Claim4: $|[S'_M/R]| \geq n$.

From Observation 37(2), we obtain $|[S'_M/R]| = |[(I \cup J) \cup (K \cup L)/R]|$. From Definition 36, and since R is a strict (S_M, S_N)-separator, we can use Observation 14(1), and write $|[S'_M/R]| = |[I \cup J/R]| + |[K \cup L/R]| \geq |[I/R]| + |[L/R]|$, the inequality being immediately obvious. Putting it together, using the first two claims, we get $|[S'_M/R]| \geq l + (n - l) = n$.

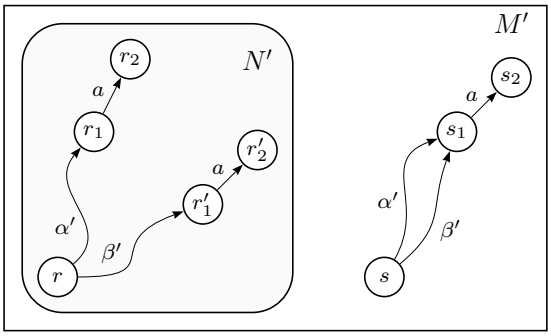

Figure 4: Lemma 40 illustration

The claims are established and the proof is complete. ∎

The next lemma says that, in the implementation, if a given additional state and a submachine initial state are R-equivalent, then the concatenation of a certain sequence with $R \cup T$ can distinguish them.

Lemma 40. *Let $s \in K^r$ and $r \in I'_N$ be two distinguishable states of M'. If $s_0 \approx_\pi s'_0$, then there exists an input sequence β that satisfies:*

1. *Either $s \not\approx_\beta r$, or $\widehat{\delta}'(\beta, s) \not\approx_{T \cup R} \widehat{\delta}'(\beta, r)$, and*
2. *With the exception of $\widehat{\delta}'(\beta, s)$, the states reached using β starting at s are all distinct and belong to K.*

Proof. Since s and r are distinguishable, there exists a sequence that conforms to the first requirement. So, let α be a minimal sequence satisfying the first property. If $\alpha = \varepsilon$, then α reaches only $s \in K$. In this case, α also satisfies the second property and we are done.

Now, assume that $\alpha \neq \varepsilon$, and let $\alpha = \alpha' a$, where $\alpha' \in X^*$ and $a \in X$. Let $s_1, s_2 \in S'$ be such that $s \xrightarrow{\alpha'} s_1 \xrightarrow{a} s_2$. From Observation 23, we also have $r_1, r_2 \in S'_N$ with $r \xrightarrow{\alpha'} r_1 \xrightarrow{a} r_2$. See Figure 4.

Claim 1: $s_1 \in S'_M$.

Suppose, for the sake of contradiction, that $s_1 \in S'_N$. Let $s_N \in I'_N$ be the first state in S'_N reached using α' starting at s, and let $\alpha = \alpha_M \alpha_N$ with $s \xrightarrow{\alpha_M} s_N$. From Observation 23, applying α_M starting at r, we reach some $r_N \in S'_N$ such that $r \xrightarrow{\alpha_M} r_N$. Let S'^{ri}_N as in Lemma 35. Since $r \in I_N$, we get $r_N \in S'^{ri}_N$. Clearly, $|\alpha_M| \leq |\alpha'| < |\alpha|$ and so, from the minimality of α, we conclude that $s_N \approx_{T \cup R} r_N$ and $s \approx_{\alpha_M} r$. Therefore, $s_N \approx_R r_N$. Recalling, we have $s_N \in I'_N$, $r_N \in S'^{ri}_N$. Since, from Lemma 35(2), R is a (I'_N, S'^{ri}_N)-separator, we get $s_N \approx r_N$. So, $s_2 \approx_{T \cup R} r_2$. Since $s_2 = \widehat{\delta}'(\alpha, s)$ and $r_2 = \widehat{\delta}'(\alpha, r)$, we may write $\widehat{\delta}'(\alpha, s) \approx_{T \cup R} \widehat{\delta}'(\alpha, r)$. Because α satisfies the first

property, we conclude that $s \not\approx_\alpha r$, and so we must have $\widehat{\lambda}'(\alpha, s) \neq \widehat{\lambda}'(\alpha, r)$. But

$$\begin{aligned}\widehat{\lambda}'(\alpha, s) &= \widehat{\lambda}'(\alpha_M, s)\widehat{\lambda}'(\alpha_N, s_N) \\ &= \widehat{\lambda}'(\alpha_M, r)\widehat{\lambda}'(\alpha_N, r_N) = \widehat{\lambda}'(\alpha, r).\end{aligned}$$

This contradiction establishes the Claim 1.

Let now β' be a sequence with minimum length such that $s \xrightarrow{\beta'} s_1$. We will show that $\beta = \beta' a$ satisfies both desired properties.

Claim 2: β satisfies property (2).

Let β'' be a prefix of β', then $|\beta''| \leq |\beta'| \leq |\alpha'| < |\alpha|$. Let states r' and s' be such that $s \xrightarrow{\beta''} s'$ and $r \xrightarrow{\beta''} r'$. From Observation 23, we get $r' \in S'_N$. Also, we have $s' \in S_M$, otherwise we would get $s_1 \in S'_N$, using Observation 23 again. By the minimality of α, we get $s' \approx_{T \cup R} r'$, and so $s' \approx_R r'$. From $r \xrightarrow{\beta''} r'$ and $r \in I'_N$, using Observation 26(2), we get $u' \in S_N$ such that $u' \approx r'$, so $s' \approx_R u'$. Since $s' \in S'_M$, $u' \in S_N$ and $s' \approx_R u'$, from Definition 36(2), we have $s' \in K$. We have thus shown that all states reached from s using β' are in K. Since β' is minimal, they are also all distinct. So, given that $\beta = \beta' a$, we conclude that β satisfies the second property.

Claim 3: β satisfies property (1).

Let r'_1 and r'_2 be states such that $r \xrightarrow{\beta'} r'_1 \xrightarrow{a} r'_2$. Recall Figure 4. From Claim 2, we know that, starting in s, β' reaches only distinct states in K. Using Lemma 39(3), we may write $|\beta'| \leq |K| - 1 \leq |K| + |J| - 1 \leq m - n - 1$. Since $r \in I'_N$, it follows that r'_1 is in the $(m-n-1)$-neighborhood of I'_N, that is, $r'_1 \in A'$. Also, since $r \in I'_N$, and $r \xrightarrow{\alpha'} r_1$, we have $r_1 \in S'^{ri}_N$. Since $|\beta'| < |\alpha|$ and $|\alpha'| < |\alpha|$, by the minimality of α again, $r'_1 \approx_{T \cup R} s_1$ and $r_1 \approx_{T \cup R} s_1$. Hence, $r'_1 \approx_{T \cup R} r_1$, with $r'_1 \in A'$ and $r_1 \in S'^{ri}_N$. Because, from Lemma 35(1), T is an (A', S'^{ri}_N)-separator, we get $r'_1 \approx r_1$. Then $r_2 \approx r'_2$.

Since α satisfies the first property, $s_2 \not\approx_{T \cup R} r_2$ or $s \not\approx_\alpha r$.

1. In the first case, take $s_2 \not\approx_{T \cup R} r_2$. We get $s_2 \not\approx_{T \cup R} r'_2$, since $r_2 \approx r'_2$. But $s_2 = \widehat{\delta}'(\beta, s)$ and $r'_2 = \widehat{\delta}'(\beta, r)$. So, $\widehat{\delta}'(\beta, s) \not\approx_{T \cup R} \widehat{\delta}'(\beta, r)$, and so β also satisfies the first property.
2. In the second case, take $s \not\approx_\alpha r$. Since $|\alpha'| < |\alpha|$, the minimality of α gives $s_1 \approx_{\alpha'} r_1$. Together with $\alpha = \alpha' a$, we conclude that $\lambda'(a, r_1) \neq \lambda'(a, s_1)$. So $\lambda'(a, r'_1) \neq \lambda'(a, s_1)$, since $r_1 \approx r'_1$. But then we obtain $s \not\approx_\beta r$, establishing that β also satisfies the first property in this case.

The proof is now complete. ∎

7.2 Obtaining a partial characterization set

The set R was defined as a separator for submachine states and additional states in the specification. Now we show that R is also a separator for submachine states and additional states in an implementation that passes the test suite.

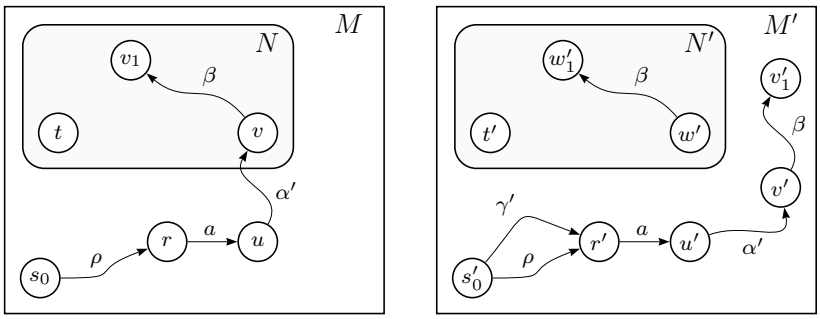

Figure 5: Lemma 41 illustration

Lemma 41. *If $s_0 \approx_\pi s'_0$, then R is a (I'_N, S'^r_M)-separator.*

Proof. Let $s' \in S'^r_M$ and $t' \in I'_N$ be two distinguishable states. It suffices to show that $s' \not\approx_R t'$.

We will depict the situation. Since $t' \in I'_N$, using Observation 26(4), we get $t \in I_N$ such that $t \approx t'$. Suppose that $s' \notin K$, then by Definition 36(2), we get $s' \not\approx_R t$, and so $s' \not\approx_R t'$, as desired. We will show that, in fact, $s' \notin K$. So, assume, for the sake of contradiction, that $s' \in K$.

From Observation 38, we have $s'_0 \in I$, and since $s' \in K$, we have $s'_0 \neq s'$, otherwise I and K would intersect, contradicting Observation 37(1). We have $s' \in S'^r_M$, then there exists a sequence γ, with minimum length, such that $s'_0 \xrightarrow{\gamma} s'$. Since $s'_0 \neq s'$, we have $|\gamma| \geq 1$.

Let r' be last state in I reached using γ from s'_0, and let $\gamma', \alpha \in X^*$, $a \in X$, and $u' \in S'^r$, such that $\gamma = \gamma' a \alpha$, and $s'_0 \xrightarrow{\gamma'} r' \xrightarrow{a} u' \xrightarrow{\alpha} s'$. Since $s' \in S'_M$, using Observation 23, we get $u' \in S'_M$. Because $r' \in I$, by Definition 36(1), there exist $r \in S_M$, $\rho \in Q$, such that $s'_0 \xrightarrow{\rho} r' \xrightarrow{a} u' \xrightarrow{\alpha} s'$, and $s_0 \xrightarrow{\rho} r \xrightarrow{a} u \xrightarrow{\alpha} s$, for some states $u, s \in S$. See Figure 5.

We have $s_0 \approx_\pi s'_0$, that is, $s_0 \approx_{P \otimes Z} s'_0$. Since $\rho a \in P$ and $\widehat{\delta}(\rho a, s_0) = u$, we get $s_0 \approx_{\rho a Z(u)} s'_0$. It follows that $u \approx_{Z(u)} u'$, and so $u \approx_Z u'$.

Clearly, starting at u', the word α does not reach any state in I, because r' is the last state in I reached using γ. With the following claims, we will show that α does not reach states in L or K.

Claim 1: α does not reach any state in L.

> For contradiction, let α' be the minimal prefix of α, such that $u' \xrightarrow{\alpha'} v'$ and $v' \in L$. Therefore, starting at u', α' reaches only distinct states in $J \cup K$ and state v'. From Lemma 39(3), $|\alpha'| \leq |J| + |K| \leq m - n$, so $\alpha' \in X_{m-n}$. Now, let $v \in S$, such that $u \xrightarrow{\alpha'} v$. From $u \approx_Z u'$, and since $Z = X_{m-n}R$, we get $u \approx_{\alpha' R} u'$, and so $v \approx_R v'$. Since $v \in S$, by Definition 36(3), we get $v' \notin L$. This contradiction estabilishes Claim 1.

Claim 2: Let $\sigma \in X^*$. Suppose that, starting at u', σ reaches only distinct states of $J \cup K$ and another state of S'. If $u \xrightarrow{\sigma} v$ and $u' \xrightarrow{\sigma} v'$, then $v \approx_{\overline{\mathcal{R}}(v)} v'$ and $v \approx_R v'$.

From Lemma 39(3), we have $|\sigma| \leq |J| + |K| \leq m - n$, so $\sigma \in X_{m-n}$. Then, from $u \approx_{\mathcal{Z}(u)} u'$, since $\mathcal{Z}(u) = X_{m-n} \underset{u}{\otimes} \mathcal{R}$ and $\widehat{\delta}(\sigma, u) = v$, we have $u \approx_{\sigma\overline{\mathcal{R}}(v)} u'$. It follows $v \approx_{\overline{\mathcal{R}}(v)} v'$. Clearly, $R \subseteq \overline{\mathcal{R}}(v)$, then we also have $v \approx_R v'$.

Claim 3: Let α' be the minimal prefix of α, such that $u' \xrightarrow{\alpha'} v'$ and $v' \in K$. If $v \in S$ such that $u \xrightarrow{\alpha'} v$, then $v \in I_N$.
From Definition 36(2), and since $v' \in K$, there must exist a state $w \in S_N$, such that $w \approx_R v'$. From Claims 1 and 2, we obtain $v \approx_R v'$, and then $v \approx_R w$. Since R is a strict (S_N, S_M)-separator and $w \in S_N$, from Lemma 17(3), $v \notin S_M$, and then $v \in S_N$. Since $r \in S_M$, we know, from Definition 24, that $u \in S_M \cup I_N$. If $v = u$, then $v \in I_N$, and we are done. So, we may assume $v \neq u$. It follows that $|\alpha'| \geq 1$, and so there exist $b \in X$ and $\alpha'' \in X^*$ such that $\alpha' = \alpha''b$. Let $u \xrightarrow{\alpha''} x$ and $u' \xrightarrow{\alpha''} x'$. From Claims 1 and 2, we obtain $x \approx_R x'$, and, from Claim 1, combined with the minimality of α', we obtain $x' \in J$. From the Definition 36(1), there exists $y \in S_M$, such that $y \approx_R x'$, and then $x \approx_R y$. Since R is a strict (S_N, S_M)-separator and $y \in S_M$, from Lemma 17(3), $x \notin S_N$, so $x \in S_M$. Since $x \xrightarrow{b} v$, from Definition 24, it follows that $v \in I_N$.

Claim 4: α does not reach any state in K.
For contradiction, let α' be the minimal prefix of α, such that $u' \xrightarrow{\alpha'} v'$ and $v' \in K$, and let $v \in S$ such that $u \xrightarrow{\alpha'} v$. From Claim 3, we have $v \in I_N$, then, from Observation 26(3), we have $w' \in I'_N$, such that $v \approx w'$. Let β be a sequence that satisfies the properties of Lemma 40 for states $v' \in K^r$ and $w' \in I'_N$. Let v_1, w'_1 and v'_1 be states such that $v \xrightarrow{\beta} v_1$, $w' \xrightarrow{\beta} w'_1$ and $v' \xrightarrow{\beta} v'_1$. Recall Figure 5.

Consider the sequence $\alpha'\beta$. From Claim 1, combined with the minimality of α', starting at u', only v' and distinct states in set J are reached using α'. Also, since β satisfies property (2) of Lemma 40, starting at v', only v'_1 and distinct states in set K are reached using β. Therefore, from Lemma 39(3), we get $|\alpha'\beta| \leq |J| + |K| \leq m - n$, so $\alpha'\beta \in X_{m-n}$.

Affirmation: $v \approx_\beta v'$ and $v_1 \approx_{R \cup T} v'_1$.
We have $u \approx_Z u'$, so $u \approx_{X_{m-n}} u'$. It follows that $u \approx_{\alpha'\beta} u'$, and then $v \approx_\beta v'$. Also, from Claim 2, we get $v_1 \approx_{\overline{\mathcal{R}}(v_1)} v'_1$. Since $v \in S_N$, from Observation 23, $v_1 \in S_N$, so $\overline{\mathcal{R}}(v_1) = \mathcal{R}(S_N) = R \cup T$. Therefore, $v_1 \approx_{R \cup T} v'_1$.

Recall that β satisfies property (1) of Lemma 40. Then $w'_1 \not\approx_{R \cup T} v'_1$ or $w' \not\approx_\beta v'$. Since $v \approx w'$, and then $v_1 \approx w'_1$, we have either $v_1 \not\approx_{R \cup T} v'_1$ or $v \not\approx_\beta v'$. However, this contradicts the Affirmation, establishing the Claim 4.

Since we supposed $s' \in K$, the Claim 4 is a contradiction. The proof is now complete. ∎

In the next corollaries, we show that R takes on the role of special kinds of separator.

Corollary 42. *If $s_0 \approx_\pi s'_0$, then R is a $(I'_N \cup S'^r_M, I'_N)$-separator.*

Proof. Let S'^{ri}_N as in Lemma 35, then R is a (I'_N, S'^{ri}_N)-separator. Also, from Lemma 41, R is a (S'^{r}_M, I'_N)-separator. Combining these with Lemma 17(2), we get $R \cup R$ is a $(I'_N \cup S'^{r}_M, S'^{ri}_N \cap I'_N)$-separator. That is, R is a $(I'_N \cup S'^{r}_M, I'_N)$-separator. ∎

Corollary 43. *If $s_0 \approx_\pi s'_0$, then R is a $(\mathrm{nbh}(S'^{r}_M, 1), \mathrm{nbh}(S'^{r}_M, 1) \setminus S'^{r}_M)$-separator.*

Proof. From Corolllary 42, R is a $(I'_N \cup S'^{r}_M, I'_N)$-separator. Since every transition that enters S'_N reaches I'_N, we know that $\mathrm{nbh}(S'^{r}_M, 1) \subseteq I'_N \cup S'^{r}_M$. Also, clearly, $\mathrm{nbh}(S'^{r}_M, 1) \setminus S'^{r}_M \subseteq I'_N$. Hence, using Lemma 17(5), we know that R is a $(\mathrm{nbh}(S'^{r}_M, 1), \mathrm{nbh}(S'^{r}_M, 1) \setminus S'^{r}_M)$-separator. ∎

Now we show that, for an implementation that passes the test suite, Z distinguishes every pair of additional reachable and distinguishable states. This is in contrast with the analogous sets defined by W-method [4] and by G-method [2], since, here, Z is meant to separate only additional states, thus being a potentially weaker separator than those analogous sets, which must separate any pair of states in the whole implementation.

Corollary 44. *If $s_0 \approx_\pi s'_0$, then Z is a partial characterization set for S'^{r}_M.*

Proof. Define $d = |[(S'_M \setminus S'^{r}_M)/R]|$, $m' = m - d$ and $n' = n - d$. We have $m' = m - d \geq |S'_M| - |[(S'_M \setminus S'^{r}_M)/R]| \geq |S'_M| - |S'_M \setminus S'^{r}_M| = |S'^{r}_M|$. Also, from Lemma 39(4) and Observation 14(4), we have $n' = n - d \leq |[S'_M/R]| - |[(S'_M \setminus S'^{r}_M)/R]| \leq |[S'^{r}_M/R]|$.

Next, we want to use Corollary 20. Let $C = S'^{r}_M$, $B = \mathrm{nbh}(S'^{r}_M, 1) = \mathrm{nbh}(C, 1)$. Then, R is a $(B, B \setminus C)$-separator. Since $n' \leq |[S'^{r}_M/R]|$, R partitions C in at least n' classes. Also, $m' \geq |S'^{r}_M|$, and so m' is an upper bound on the number of classes of equivalence in C. Now, using Corollary 20, we know that $X_{m'-n'}R$ is a partial characterization set for S'^{r}_M. Finally, note that $m' - n' = m - n$. Then $Z = X_{m-n}R = X_{m'-n'}R$ is a partial characterization set for S'^{r}_M. ∎

7.3 Generating a complete test suite

The next lemma states that, for each reached state in the specification, there exists a corresponding Z-equivalent state in an implementation that passes the test suite.

Lemma 45. *Suppose $s_0 \approx_\pi s'_0$. Let $\gamma \in X^*$ and let $s \in S$, $s' \in S'$ be such that $s_0 \xrightarrow{\gamma} s$ and $s'_0 \xrightarrow{\gamma} s'$. Then $s \approx_Z s'$.*

Proof. For the sake of contradiction, let $\gamma \in X^*$ be minimal such that $s_0 \xrightarrow{\gamma} s$, $s'_0 \xrightarrow{\gamma} s'$ and $s \not\approx_Z s'$. From Observation 38, we know that $s_0 \approx_Z s'_0$, then $|\gamma| > 0$. Let $\gamma' \in X^*, b \in X$ be such that $\gamma = \gamma' b$. Let r be the last state in S_M that can be reached using γ' and starting at s_0. Take $\gamma_1, \gamma_2 \in X^*$ such that $\gamma = \gamma_1\gamma_2$ and $s_0 \xrightarrow{\gamma_1} r$. Note that γ_1 is the maximal prefix of γ' such that $r \in S_M$. Let also $r' \in S'$ be such that $s'_0 \xrightarrow{\gamma_1} r'$. Since $|\gamma_2| > 0$, there exist $\alpha \in X^*, a \in X$ such that $\gamma_2 = a\alpha$. Let $u \in S, u' \in S'$ be such that $r \xrightarrow{a} u$ and $r' \xrightarrow{a} u'$. We now have

$$s_0 \xrightarrow{\gamma_1} r \xrightarrow{a} u \xrightarrow{\alpha} s \qquad \text{and} \qquad s'_0 \xrightarrow{\gamma_1} r' \xrightarrow{a} u' \xrightarrow{\alpha} s'.$$

CLAIM 1: $u \approx_Z u'$.

Since $r \in S_M$, we know that there exist $\rho, \rho a \in P$ such that $s_0 \xrightarrow{\rho} r$, and there exist $r'', u'' \in S'$ such that $s'_0 \xrightarrow{\rho} r'' \xrightarrow{a} u''$. Since $s_0 \approx_{PZ} s'_0$, we have $r \approx_Z r''$ and $u \approx_Z u''$. From the minimality of γ, we also have $r \approx_Z r'$. Then $r' \approx_Z r''$. Since $R \subseteq Z$, we get $r \approx_R r'$ and $r \approx_R r''$. From Lemma 34, $r', r'' \in S'_M$, and then $r', r'' \in S'^r_M$. From Corollary 44, Z is a partial characterization set of S'^r_M. It follows that $r' \approx r''$, and so $u' \approx u''$. Since we already have $u \approx_Z u''$, we get $u \approx_Z u'$.

CLAIM 2: $s \approx_Z s'$.

Let $|\alpha| = 0$, then $u = s$ and $u' = s'$. From CLAIM 1, we obtain $s \approx_Z s'$, and we are done. Now, assume $|\alpha| > 0$, then we conclude that γ' reaches u. We must have $u \notin S_M$, otherwise γ_1 would not be maximal. Since $r \in S_M$, we also obtain $u \in I_N$, using Definition 24. From Observation 26(3), there exists $t' \in I'_N$ such that $u \approx t'$. Since, from CLAIM 1, we already have $u \approx_Z u'$, we get $t' \approx_Z u'$, and so $t' \approx_R u'$.

Since $r' \in S'^r_M$, we get $u' \in I'_N \cup S'^r_M$. We now have $t' \approx_R u'$, with $t' \in I'_N$ and $u' \in I'_N \cup S'^r_M$. From Corollary 42, R is a $(I'_N, I'_N \cup S'^r_M)$-separator, then it follows that $t' \approx u'$. Since we already have $u \approx t'$, we get $u \approx u'$, and then $s \approx_Z s'$.

The CLAIM 2 is a contradiction, so the proof is complete. ∎

The next result states that an incorrect implementation, we cannot have Z-equivalence between states in the specification and in the implementation.

Lemma 46. *Suppose $s_0 \approx_\pi s'_0$. If $s_0 \not\approx s'_0$, then there exist $\gamma \in X^*$, $s \in S$ and $s' \in S'$ such that $s_0 \xrightarrow{\gamma} s$, $s'_0 \xrightarrow{\gamma} s'$, and $s \not\approx_Z s'$.*

Proof. Let $\gamma \in X^*$ be minimal such that $s_0 \not\approx_\gamma s'_0$. Since $|\gamma| > 0$, there exist $\gamma' \in X^*, b \in X$ such that $\gamma = \gamma' b$. Let r be the last state in S_M that can be reached using γ' and starting at s_0. Take $\gamma_1, \gamma_2 \in X^*$ such that $\gamma = \gamma_1\gamma_2$ and $s_0 \xrightarrow{\gamma_1} r$. Note that γ_1 is the maximal prefix of γ' such that $r \in S_M$. Let also $r' \in S'$ be such that $s'_0 \xrightarrow{\gamma_1} r'$. Since $|\gamma_2| > 0$, there exist $\alpha \in X^*, a \in X$ such that $\gamma_2 = a\alpha$. Since $r \in S_M$, we know that there exist $\rho, \rho a \in P$ such that $s_0 \xrightarrow{\rho} r$. Let $u \in S$, and $u', r'', u'' \in S'$ be such that $r \xrightarrow{a} u$, $r' \xrightarrow{a} u'$, and $s'_0 \xrightarrow{\rho} r'' \xrightarrow{a} u''$. We now have

$$s_0 \xrightarrow{\rho,\gamma_1} r \xrightarrow{a} u \xrightarrow{\alpha} s, \qquad s'_0 \xrightarrow{\gamma_1} r' \xrightarrow{a} u' \xrightarrow{\alpha} s', \qquad \text{and} \qquad s'_0 \xrightarrow{\rho} r'' \xrightarrow{a} u''.$$

We will show that $r \not\approx_Z r'$. Suppose, for the sake of contradiction, that $r \approx_Z r'$.

CLAIM 1: $r \approx r''$ and $u \approx_Z u'$.

Since $s_0 \approx_{PZ} s'_0$, we have $r \approx_Z r''$ and $u \approx_Z u''$. Since $r \approx_Z r'$, we also have $r' \approx_Z r''$. Since $R \subseteq Z$, we get $r \approx_R r'$ and $r \approx_R r''$. From Lemma 34, $r', r'' \in S'_M$, and then $r', r'' \in S'^r_M$. From Corollary 44, Z is a partial characterization set of S'^r_M. It follows that $r' \approx r''$, and so $u' \approx u''$. Since we already have $u \approx_Z u''$, we get $u \approx_Z u'$.

CLAIM 2: $s_0 \approx_\gamma s'_0$.

Since $s_0 \approx_P s'_0$ and $\rho a \in P$, we get $\widehat{\lambda}(\rho a, s_0) = \widehat{\lambda}'(\rho a, s'_0)$. Then, $\lambda(a, r) = \lambda'(a, r'')$,

and so $\lambda(a,r) = \lambda'(a,r')$, since, from CLAIM 1, we have $r'' \approx r'$. From the minimality of γ, we also have $s_0 \approx_{\gamma_1} s'_0$, and so we get $s_0 \approx_{\gamma_1 a} s'_0$. Let $|\alpha| = 0$, then $\gamma = \gamma_1 a$. It follows that $s \approx_\gamma s'$, and we are done. Now, assume $|\alpha| > 0$, then we conclude that γ' reaches u. We must have $u \notin S_M$, otherwise γ_1 would not be maximal. Since $r \in S_M$, we also obtain $u \in I_N$, using Definition 24. From Observation 26(3), there exists $t' \in I'_N$ such that $u \approx t'$. Since, from CLAIM 1, we already have $u \approx_Z u'$, we get $t' \approx_Z u'$, and so $t' \approx_R u'$.

Since $r' \in S'^r_M$, we get $u' \in I'_N \cup S'^r_M$. We now have $t' \approx_R u'$, with $t' \in I'_N$ and $u' \in I'_N \cup S'^r_M$. From Corollary 42, R is a $(I'_N, I'_N \cup S'^r_M)$-separator, then it follows that $t' \approx u'$. Since we already have $u \approx t'$, we get $u \approx u'$, and then $u \approx_\alpha u'$. Since we also have $s_0 \approx_{\gamma_1 a} s'_0$, we get $s_0 \approx_{\gamma_1 a \alpha} s'_0$, and so $s_0 \approx_\gamma s'_0$.

The CLAIM 2 is a contradiction, so we get $r \not\approx_Z r'$. Now, we have $\gamma_1 \in X^*$, $r \in S$, $r' \in S'$ and $s_0 \xrightarrow{\gamma_1} r$, $s'_0 \xrightarrow{\gamma_1} r'$ with $r \not\approx_Z r'$, as desired. This completes the proof. ∎

The last two lemmas lead to our main result: a combined candidate implementation is equivalent to the combined specification if and only if the implementation passes the given test suite.

Theorem 47. *$s_0 \approx s'_0$ if and only if $s_0 \approx_\pi s'_0$.*

Proof. If $s_0 \approx s'_0$, then clearly $s_0 \approx_\pi s'_0$.

Now assume $s_0 \approx_\pi s'_0$. For the sake of contradiction, assume $s_0 \not\approx s'_0$. By Lemma 46, we get some $\gamma \in X^*$ with $s_0 \xrightarrow{\gamma} s$, $s'_0 \xrightarrow{\gamma} s'$ and $s \not\approx_Z s'$. Given that, Lemma 45 gives $s \approx_Z s'$. This contradiction forces $s_0 \approx s'_0$, as desired. ∎

8 Concluding Remarks

Model-based testing methods are widely used to test critical systems. The well-known W-method [4] is one among such methods. It can be used to test *completely specified* and *deterministic* FSMs. However, if the number of states is large, then the W-method, and variations of it, become impractical. Moreover, in several common situations, using the W-method is inefficient. Such cases include testing implementations that were modified after minor specifications changes have been applied. Another example stems from testing systems developed using a building block strategy.

To address these issues, in this paper, we considered a FSM as a composition of other previously tested submachines. This gave rise to the concept of combined FSMs. With this new concept, we were able to represent the above situations in a specific fault model, for which we adapted the W-method. This resulted in a new test method for combined FSMs, called the C-method, which can generate much smaller test suites.

We also introduced the concept of separators, generalizing the notions of characterization sets and identification sets, which were used in the W-method and in the Wp-method, respectively. Separators showed to be useful tools to test FSM. For example, in order to reduce the test suite generated using the W-method, we showed how to replace the

characterization set by a smaller separator. The greatest advantage of using separators is that we can use them to distinguish as few states as we need, therefore resulting in smaller sets of distinguishing sequences.

We also adapted some ideas from the G-method [2]. In fact, it turns out that the G-method is a particular case of our method when the set of submachines is empty. It has been shown that for special families of naturally occurring FSMs, the gain of G-method is exponential [3], given some information, supplied by the user, about the set of implementation candidates. We showed that, even if this information is not available, we may still extract parameters from the specification and proceed to generate complete test suites.

Finally, we showed that the ratio between the size of the number of test cases generated using the W-method and the number of test cases generated using the C-method grows exponentially, as we increase the number of submachine states. This lead to the conclusion that the C-method is scalable, provided that the number of additional states is kept small. We conclude that we can *test a FSM with an arbitrarily large number of states*, if we use the proposed building block strategy.

References

[1] Gregor V. Bochmann and Alexandre Petrenko. Protocol testing: review of methods and relevance for software testing. In *ISSTA '94: Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis*, pages 109–124, 1994.

[2] Adilson Luiz Bonifácio, Arnaldo Vieira Moura, and Adenilso da Silva Simão. A generalized model-based test generation method. In *6th IEEE International Conferences on Software Engineering and Formal Methods*, pages 139–148, 2008.

[3] Adilson Luiz Bonifácio, Arnaldo Vieira Moura, and Adenilso da Silva Simão. Exponentially more succinct test suites. Technical Report IC-09-07, Institute of Computing, University of Campinas, 2009.

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

[5] Edmund M. Clarke and Jeannette M. Wing. Formal methods: state of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[6] R. Dorofeeva, Khaled El-Fakih, and Nina Yevtushenko. An improved conformance testing method. In *IFIP 25th International Conference on Formal Techniques for Networked and Distributed Systems*, pages 204–218, 2005.

[7] Khaled El-Fakih, Vadim Trenkaev, Natalia Spitsyna, and Nina Yevtushenko. Fsm based interoperability testing methods for multi stimuli model. In *Testing of Communicating Systems*, pages 60–75, 2004.

[8] Khaled El-Fakih, Nina Yevtushenko, and Gregor von Bochmann. Fsm-based re-testing methods. In *TestCom '02: Proceedings of the IFIP 14th International Conference on Testing Communicating Systems XIV*, pages 373–390, 2002.

[9] Khaled El-Fakih, Nina Yevtushenko, and Gregor von Bochmann. Fsm-based incremental conformance testing methods. *IEEE Transactions on Software Engineering*, 30(7):425–436, 2004.

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

[11] Arthur Gill. *Introduction to the theory of finite state machines*. McGraw-Hill, 1962.

[12] Koufareva, Alexandre Petrenko, and Nina Yevtushenko. Test generation driven by user-defined fault models. In *Proceedings of the IFIP TC6 12th International Workshop on Testing Communicating Systems*, pages 215–236, 1999.

[13] David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines - a survey. In *Proceedings of the IEEE*, pages 1090–1123, 1996.

[14] Gang Luo, Gregor V. Bochmann, and Alexandre Petrenko. Test selection based on communicating nondeterministic finite-state machines using a generalized wp-method. *IEEE Transactions on Software Engineering*, 20(2):149–162, 1994.

[15] Gang Luo, Alexandre Petrenko, and Gregor V. Bochmann. Selecting test sequences for partially-specified nondeterministic finite state machines. In *IFIP 7th International Workshop on Protocol Test Systems*, pages 91–106, 1994.

[16] Glenford J. Myers. *Art of Software Testing*. John Wiley & Sons, Inc., 2004.

[17] Alexandre Petrenko. Fault model-driven test derivation from finite state models: annotated bibliography. In *Modeling and verification of parallel processes*, volume 2067, pages 196–205, 2001.

[18] Patrick J. Schroeder and Bogdan Korel. Black-box test reduction using input-output analysis. *SIGSOFT Software Engineering Notes*, 25(5):173–177, 2000.

[19] D.P. Sidhu and T.-K. Leung. Formal methods for protocol testing: a detailed study. *IEEE Transactions on Software Engineering*, 15(4):413–426, 1989.