Adaptive Testing of Deterministic Implementations Specified by Nondeterministic FSMs

Alexandre Petrenko [+] and Nina Yevtushenko [*]

[+] CRIM, Centre de recherche informatique de Montréal, 405 Ogilvy Avenue, Suite 101
Montréal (Québec) H3N 1M3, Canada
[*] Tomsk State University, 36 Lenin Street, Tomsk, 634050, Russia
petrenko@crim.ca; ninayevtushenko@yahoo.com

Abstract. The paper addresses the problem of adaptive testing of a deterministic FSM which models an implementation under test using a nondeterministic FSM as its specification. It elaborates a method for deriving test fragments, combining and executing them in adaptive way such that the implementation passes the test if and only if it is a reduction of the specification. Compared to the existing methods, it uses adaptive test fragments needed to reach as well as to distinguish states.

Keywords: nondeterministic FSM, conformance testing, test generation, adaptive testing

1 Introduction

There exists a significant body of work devoted to the development of methods for test generation from a given FSM to guarantee the “full” fault coverage. Such coverage of resulting tests means the following. Given a specification machine with n states and a fault domain containing all FSMs with at most m states, $m \geq n$, the full fault coverage is achieved by a so-called m-complete test suite, which detects all faults in any implementation that can be modelled by an FSM in the fault domain. An implementation has a fault if it does not respect a chosen conformance relation, typically trace equivalence or inclusion. To derive m-complete tests the existing methods (with reset operation) use the following three test fragments:

- transfer sequences/strategies to reach states in the specification FSM;
- traversal sequences to extend the transfer sequences; in case of $m = n$ they ensure the transition coverage of the specification and implementation machines and in case of $m > n$ additionally check for the existence of extra states in the implementation machines;
- state identification or distinguishing sequences/strategies to check states reached by prefixes of the above sequences.

In the case of deterministic specifications, several methods are elaborated, such as the W, Wp, HSI, H, and more recently the SPY method [10]. While differing in the types of state identifiers, they require the same traversal set of all input sequences of length $m - n + 1$ be applied after each state of the specification. However, as the results of [9] show, different traversal sets should be used when the specification has undistinguishable states. Moreover, it is no longer required to reach with transfer sequences each and every state of such specifications.

When the specification can be nondeterministic and a conforming implementation FSM is its reduction, it can have fewer traces than the specification, and not all the states of the specification can be matched with the states of the implementation. The implication for deriving m-complete test suites is that the value of m can even be smaller than that of n. This fact has to be taken into account in determining each of the three test fragments and the way they are composed to yield m-complete test for a nondeterministic FSM. State reachability and distinguishability can be achieved in testing more efficiently using adaptive execution of inputs, where the choice of a next input depends on a current output, as early work of [1, 8] indicates.

The main contribution of this paper is a method for deriving test fragments, combining and executing them in an adaptive manner against a given deterministic implementation FSM with at most m states such that the resulting verdict is pass if and only if the implementation is a reduction of the specification. The method allows to avoid the derivation of preset m-complete tests, as they could be voluminous. At the same time, we prove that the latter is the union of the tests executed for each implementation with at most m states.

The remaining of this paper is organized as follows. Section 2 defines the basic notions for state machines. Section 3 explains how the test fragments for reaching and distinguishing states as well as traversal sets can be derived for a given nondeterministic FSM and defines an m-complete test as an FSM. Section 4 details the method for adaptive testing. The proposed method is illustrated in details using an example. The related work is discussed in Section 5 and Section 6 concludes the paper.

2 General Definitions

A *Finite State Machine* (FSM) $\mathcal{S}$ is a 5-tuple $(S, s_0, I, O, h_{\mathcal{S}})$, where S is a finite set of states with the initial state s_0; I and O are finite non-empty disjoint sets of inputs and outputs, respectively; $h_{\mathcal{S}}$ is a transition relation $h_{\mathcal{S}} \subseteq S \times I \times O \times S$, where a 4-tuple $(s, i, o, s') \in h_{\mathcal{S}}$ is a transition.

Sometimes we will consider instead of s_0 another state s of $\mathcal{S}$ as the initial state; such FSM $(S, s, I, O, h_{\mathcal{S}})$ is denoted $\mathcal{S}/s$.

Input sequence $\alpha \in I^*$ is a *defined* input sequence in state s of $\mathcal{S}$ if it labels a sequence of transitions starting in state s. A *trace* of $\mathcal{S}$ in state s is a string of input-output pairs which label a sequence of transitions starting in state s. Let $Tr(\mathcal{S}/s)$ or $Tr_{\mathcal{S}}(s)$ denote the set of all traces of $\mathcal{S}$ in state s, while $Tr(\mathcal{S})$ or $Tr_{\mathcal{S}}$ denote the set of traces of $\mathcal{S}$ in the initial state. Given a sequence $\beta \in (IO)^*$, we use $Pref(\beta)$ to denote the set of all prefixes of β which are in the set $(IO)^*$, and let $Pr(\beta)$ denote $Pref(\beta) \setminus \{\varepsilon\}$. Given sequence $\beta \in (IO)^*$, the *input* (*output*) *projection* of β, denoted $\beta_{\downarrow I}$ ($\beta_{\downarrow O}$), is a sequence obtained from β by erasing symbols in O (I).

We define various types of machines as follows.

FSM $\mathcal{S} = (S, s_0, I, O, h_{\mathcal{S}})$ is

- *trivial* if $h_{\mathcal{S}} = \varnothing$;
- *completely specified* (a complete FSM) if for each pair $(s, i) \in S \times I$ there exists $(o, s') \in O \times S$ such that $(s, i, o, s') \in h_{\mathcal{S}}$;
- *partially specified* (a partial FSM) if for some pair $(s, i) \in S \times I$, input i is undefined in state s, i.e., $(s, i, o, s') \notin h_{\mathcal{S}}$ for all $(o, s') \in O \times S$;
- *deterministic* (DFSM) if for each pair $(s, i) \in S \times I$ there exists at most one transition $(s, i, o, s') \in h_{\mathcal{S}}$ for some $(o, s') \in O \times S$;
- *nondeterministic* (NFSM) if for some pair $(s, i) \in S \times I$, there exist at least two transitions (s, i, o_1, s_1), $(s, i, o_2, s_2) \in h_{\mathcal{S}}$, such that $o_1 \neq o_2$ or $s_1 \neq s_2$;
- *observable* if for each two transitions (s, i, o, s_1), $(s, i, o, s_2) \in h_{\mathcal{S}}$ it holds that $s_1 = s_2$;
- *single-input* if in each state there is at most one defined input, i.e., if for each two transitions (s, i_1, o_1, s_1), $(s, i_2, o_2, s_2) \in h_{\mathcal{S}}$ it holds that $i_1 = i_2$;
- *output-complete* if for each pair $(s, i) \in S \times I$ such that the input i is defined in the state s, there exists a transition from s with i for every output;
- *acyclic* if $Tr_{\mathcal{S}}$ is finite.

Given input sequence α defined in state s, let $out_{\mathcal{S}}(s, \alpha)$ denote the set of output sequences which can be produced by $\mathcal{S}$ in response to α at state s, that is $out_{\mathcal{S}}(s, \alpha) = \{\beta_{\downarrow O} \mid \beta \in Tr(\mathcal{S}/s)$ and $\beta_{\downarrow I} = \alpha\}$. Given an observable FSM $\mathcal{S}$, for a trace $\beta \in Tr(\mathcal{S}/s)$, s-after-β denotes the state reached by $\mathcal{S}$ when it executes the trace β from state s. If s is the initial state s_0 then instead of s_0-after-β we write $\mathcal{S}$-after-β. The FSM $\mathcal{S}$ is *initially connected*, iff for any state $s \in S$ there exists a trace β such that $\mathcal{S}$-after-$\beta = s$. A state is a *deadlock* state if no input is defined in the state and trace $\beta \in Tr(\mathcal{S})$ is a *completed* trace of $\mathcal{S}$ if $\mathcal{S}$-after-β is a deadlock state.

In this paper, we consider only complete initially connected observable specification machines; one could use a standard procedure for automata determinization to convert a given FSM into observable one. We define in terms of traces several relations between states of a complete FSM.

Given states s_1, s_2 of a complete FSM $\mathcal{S} = (S, s_0, I, O, h_{\mathcal{S}})$,

- s_1 and s_2 are (*trace-*) *equivalent*, if $Tr_{\mathcal{S}}(s_1) = Tr_{\mathcal{S}}(s_2)$;
- s_2 is *trace-included* into (is a *reduction* of) s_1, $s_2 \leq s_1$, if $Tr_{\mathcal{S}}(s_2) \subseteq Tr_{\mathcal{S}}(s_1)$;
- s_1 and s_2 are *r-compatible*, $s_1 \simeq s_2$, if there exists a state of a complete FSM that is a reduction of both states s_1 and s_2;
- s_1 and s_2 are *r-distinguishable*, $s_1 \not\simeq s_2$ if no state of any complete FSM can be a reduction of both states s_1 and s_2.

We also use relations between machines. Given FSMs $\mathcal{S} = (S, s_0, I, O, h_{\mathcal{S}})$ and $\mathcal{P} = (P, p_0, I, O, h_{\mathcal{P}})$, FSM $\mathcal{P}$ is a *reduction* of $\mathcal{S}$ if $Tr_{\mathcal{P}}(p_0) \subseteq Tr_{\mathcal{S}}(s_0)$; FSM $\mathcal{P}$ is a *submachine* of $\mathcal{S}$ if $P \subseteq S$, $p_0 = s_0$ and $h_{\mathcal{P}} \subseteq h_{\mathcal{S}}$.

To characterize the common behavior of two machines (states) we use the operation of the intersection. The *intersection* $\mathcal{S} \cap \mathcal{P}$ of two machines $\mathcal{S}$ and $\mathcal{P}$ (also known as the product) is an FSM $(Q, q_0, I, O, h_{\mathcal{S}\cap\mathcal{P}})$ with the state set $Q \subseteq S \times P$, the initial state $q_0 = s_0p_0$, and the transition relation $h_{\mathcal{S}\cap\mathcal{P}}$ such that Q is the smallest state set obtained by using the rule $(sp, i, o, s'p') \in h_{\mathcal{S}\cap\mathcal{P}} \Leftrightarrow (s, i, o, s') \in h_{\mathcal{S}}$ & $(p, i, o, p') \in h_{\mathcal{P}}$. The intersection FSM $\mathcal{S} \cap \mathcal{P}$ preserves only common traces of the two machines, in other words, for each state sp of $\mathcal{S} \cap \mathcal{P}$ we have $Tr_{\mathcal{S}\cap\mathcal{P}}(sp) = Tr_{\mathcal{S}}(s) \cap Tr_{\mathcal{P}}(p)$ and thus, $Tr_{\mathcal{S}\cap\mathcal{P}} = Tr_{\mathcal{S}} \cap Tr_{\mathcal{P}}$.

3 Deriving Test Fragments

In this section, we first establish properties of states of a specification NFSM that can be reached in an adaptive way and present a method for deriving FSM models, called state preambles, for adaptive strategies needed to reach such states. Then we consider adaptive distinguishability of states and provide a precise characterization of all possible strategies by an FSM, called a canonical separator, obtained from a self-product of the specification machine. State separators are submachines of a canonical separator. We also explain how traversal sets are derived for a given NFSM and conclude by defining a test as an FSM and its completeness.

3.1 State Preambles

All the existing test generation methods rely on tests which reach the states of the specification and match the states of the implementation and specification machines. Each such test for DFSM is completely defined by an input sequence. Once a specification is an NFSM and an implementation is allowed to have fewer traces than the specification not all the states of the specification can be matched with the states of the implementation. Indeed, a reduction of the specification FSM may have fewer states. This observation leads to the question which state of the specification FSM must be "implemented" in any correct implementation? Intuitively, it is a state such that any reduction of the specification FSM should have a trace which takes the specification FSM from the initial state to the state in question. This intuition is reflected in the following definition.

Definition 1. *Given an FSM $\mathcal{S} = (S, s_0, I, O, h_{\mathcal{S}})$, state $s \in S$ is* definitely reachable *if any reduction of $\mathcal{S}$ has a trace which takes $\mathcal{S}$ into the state s.*

This property can be established as follows.

Proposition 1. *State s of an FSM $\mathcal{S}$ is definitely reachable if and only if $\mathcal{S}$ has a single-input acyclic submachine $\mathcal{S}'$ with the only deadlock state s such that for each input defined in some state of $\mathcal{S}'$, the state has all the transitions of $\mathcal{S}$ labeled with this input.*

Such a submachine can be used in testing to adaptively bring a given machine into a definitely reachable state and is called a preamble for that state.

Definition 2. *Given a definitely reachable state $s \in S$, a single-input acyclic submachine of $\mathcal{S}$ with the only deadlock state s such that for each input defined in some state of the submachine, the state has all the outgoing transitions of $\mathcal{S}$ labeled with this input is a* preamble *for state s, denoted $\mathcal{P}_s = (R, r_0, I, O, h_{\mathcal{P}_s})$.*

A state for which there exists a preamble with a single input projection for all completed traces was called deterministically reachable state in [8]. In a deterministic (initially connected) machine each state is deterministically reachable; any nondeterministic machine has at least one deterministically, thus definitely, reachable, state, namely, the initial state.

We present a method to check whether state s is definitely reachable and, if it is, to derive a preamble $\mathcal{P}_s$.

Algorithm 1 for constructing a preamble for a given state.

Input: An FSM $\mathcal{S}$ and $s \in S$, $s \neq s_0$.

Output: a preamble if the state s is definitely reachable.

```
Construct an FSM $(R, r_0, I, O, h_{\mathcal{R}_s})$ as follows
$R := \{s\}$;
$h_{\mathcal{R}_s} := \emptyset$;
While there exist a state $s' \notin R$ and a set of inputs $I_{s'}$, such that for each input $i \in I_{s'}$, $(s', i, o, s'') \in h_{\mathcal{S}}$, $s'' \in R$ for all $o \in out_{\mathcal{S}}(s', i)$
        $R := R \cup \{s'\}$;
        $h_{\mathcal{R}_s} := h_{\mathcal{R}_s} \cup \{(s', i, o, s'') \mid i \in I_{s'}$ and $o \in out_{\mathcal{S}}(s', i)\}$;
End While;
If $s_0 \notin R$ then return the message "the state $s$ is not definitely reachable" and stop
Else let $(R, r_0, I, O, h_{\mathcal{R}_s})$, where $r_0 := s_0$, be the obtained FSM;
Starting from the initial state, remove from each state with several defined inputs all
    outgoing transitions with the same input until each state has a single defined input thus
    to obtain a single-input submachine with the only deadlock state $s$;
Delete states which are unreachable from the initial state;
Return the obtained machine as a preamble for the state $s$ and stop. ♦
```

The idea is that analyzing the backward reachability, we first choose all inputs which may form a preamble, since at that stage we do not know those leading to the required state from the initial state and then analyzing the forward reachability, we retain just a single input at each state. Since any preamble is a submachine of the specification machine with n states, then the length of any trace in a preamble never exceeds the number of states of the specification; in other words, one needs at most n - 1 inputs to transfer to a definitely reachable state.

Given a preamble $\mathcal{P}_s$, let $T(\mathcal{P}_s)$ be the set of its completed traces. Given a set K of definitely reachable states of the specification FSM $\mathcal{S}$ such that the initial state $s_0 \in K$, the union U_K of all the completed traces over all the preambles of these states is a *cover* for K.

The cover will be used in constructing another test fragment, traversal sets, as explained in Section 3.3.

Example. Consider the FSM $\mathcal{S}$ in Figure 1(a). It has four states 1, 2, 3, and 4 with state 1 as the initial state; three inputs *a*, *b*, and *c*; and two outputs 0 and 1.

The initial state is deterministically reachable, it is reached with the empty input sequence, so its preamble is a trivial FSM. Each state is definitely reachable, thus the set *K* contains all the states. Figures 2 and 3 present preambles and intermediate machines constructed using the above given method. A cover for all the states is the union of all completed traces of the obtained preambles, $U_K = \{\varepsilon, a1, a0c1b1, c0, c1a1, c1, c0c1\}$. ♦

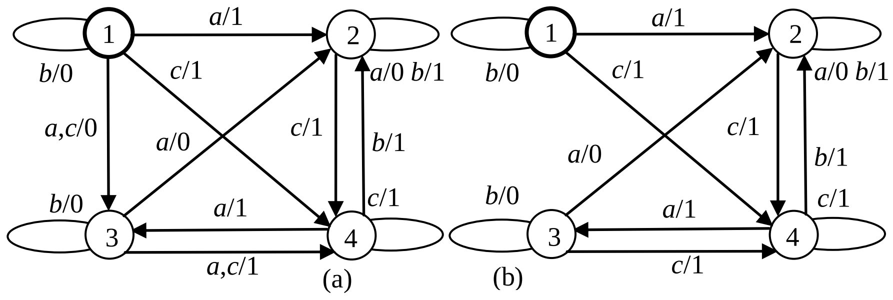

Fig. 1. (a) FSM $\mathcal{S}$ and (b) FSM $\mathcal{B}$ (initial states are in bold).

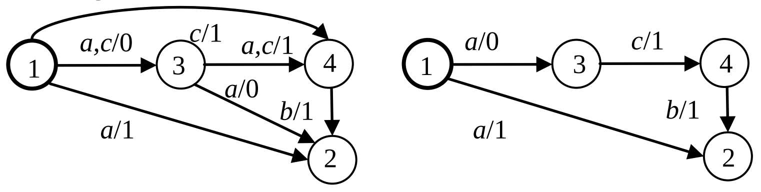

Fig. 2. Constructing a preamble for state 2.

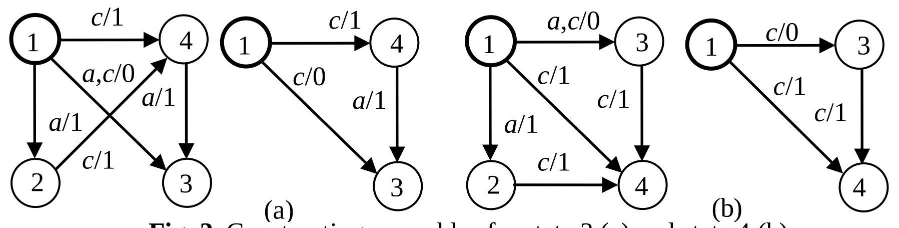

Fig. 3. Constructing preambles for state 3 (a) and state 4 (b).

3.2 State Separators

By the definition, if two states of a given machine are *r*-compatible, there exists a state of some complete FSM which is a reduction of both states. Such a state is the initial state of the intersection of two instances of a given machine initialized in different states (a self-product), since the intersection represents all the common traces of the two states. On the other hand, if the two states are *r*-distinguishable, the intersection is not a complete FSM. This fact is stated in the following.

Proposition 2. *Given two states s_1 and s_2 of a complete FSM $\mathcal{S} = (S, s_0, I, O, h_{\mathcal{S}})$ and the intersection $\mathcal{S}/s_1 \cap \mathcal{S}/s_2 = (Q, s_1s_2, I, O, h_{\mathcal{S}/s_1 \cap \mathcal{S}/s_2})$, states s_1 and s_2 are r-compatible if and only if the intersection has a complete submachine.*

Corollary 1. *States s_1 and s_2 are r-distinguishable if and only if the intersection has no complete submachine, i.e., each submachine has an input undefined in some state.*

The existence of a complete submachine can be checked by iterative removal from the intersection each state that has undefined input along with its incoming transitions. If at the end, the initial state is also removed then the two given states are *r*-distinguishable, otherwise they are *r*-compatible. The procedure is similar to the one for checking the existence of an adaptive (s_1, s_2)-distinguishing strategy considered in [1]. While that work focuses on the existence of such a strategy, it does not provide a means to characterize all the strategies and a method to obtain one. Such a method was first elaborated in [8], and now we offer an exact characterization of all state distinguishing strategies in the form of an FSM, called a canonical separator, which is obtained from the intersection as follows.

Definition 3. *Given r-distinguishable states s_1 and s_2 of an FSM $\mathcal{S}$ and the intersection $\mathcal{S}/s_1 \cap \mathcal{S}/s_2 = (Q, s_1s_2, I, O, h_{\mathcal{S}/s_1 \cap \mathcal{S}/s_2})$, an FSM $\mathcal{P} = (P, s_1s_2, I, O, h_{\mathcal{P}})$ such that $P = Q \cup \{s_1, s_2\}$ and $h_{\mathcal{P}} = h_{\mathcal{S}/s_1 \cap \mathcal{S}/s_2} \cup \{(ss', i, o, s_1) \mid ss' \in Q, o \in out(s, i) \setminus out(s', i)\} \cup \{(ss', i, o, s_2) \mid ss' \in Q, o \in out(s', i) \setminus out(s, i)\}$, is a* canonical separator *of states s_1 and s_2.*

In other words, a canonical separator for two *r*-distinguishable states s_1 and s_2 of $\mathcal{S}$ is the intersection extended by including two designated deadlock states s_1 and s_2 such that the completed traces distinguish the two possible initial states of $\mathcal{S}$.

A canonical separator contains all the traces which separate *r*-distinguishable states. For testing, it is sufficient to use acyclic traces which do not branch on inputs. This leads to the following definition.

Definition 4. *Given r-distinguishable states s_1 and s_2 of an FSM $\mathcal{S}$, a single-input acyclic submachine of the canonical separator, such that the only deadlock states are s_1 and s_2 and for each input defined in some state of the submachine, the state has all the outgoing transitions of the canonical separator labeled with this input is a* separator *of states s_1 and s_2, denoted $\mathcal{R}(s_1, s_2)$.*

By the definition, canonical separator for given states is unique, but it may contain several separators as its submachines. The procedure for determining a separator from a given canonical one is similar to Algorithm 1; it includes backward analysis and iterative removal of all defined inputs, but one for each state, as well as cycles such that the deadlock states of the submachine are reachable from all the other states and is omitted here.

Example. Figure 4(a) shows a fragment of the canonical separator of states 1 and 3 for the FSM $\mathcal{S}$ in Figure 1 (we do not show the part which starts in state 44, as states 1 and 3 cannot be reached from it). Figure 4(b) shows a separator obtained from the canonical one. The separators for other states are shown in Figure 4(c). ♦

Notice that the intersection $\mathcal{S}/s_1 \cap \mathcal{S}/s_2$ has no more than n^2 states, if $\mathcal{S}$ is an observable machine with n states. Then the length of any trace in a separator never exceeds this bound; in other words, one needs at most n^2 inputs to adaptively distinguish two states.

We can use separators of pairs of states to identify a given state among a set of possible states. Given a subset of states $L \subseteq S$, let $Id(s, L) = \{\mathcal{R}(s, s') \mid s' \in L$ and $s' \not\geq s\}$. The set $Id(s, L)$ is in some sense a generalization of a concept of a state identifier in a subset of states used in testing from deterministic FSMs. More precisely, the set $Id(s, L)$ allows one to identify that a given state cannot be a reduction of any state in the set L but state s. Such sets will be used in the main algorithm in Section 4.

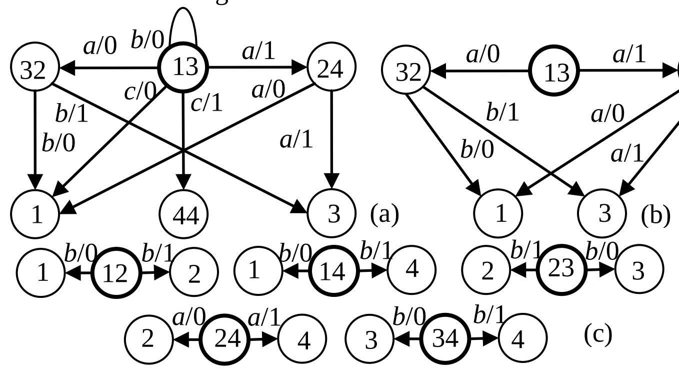

Fig. 4. (a) a fragment of the canonical separator of states 1 and 3, (b) the separator of states 1 and 3, (c) the separators for other states.

3.3 Traversal Sets

The construction of traversal sets is based on the approach elaborated in [6] and we refer the reader to that work for more detail. The basic idea is to count states of the specification FSM traversed by a trace and to terminate the trace as soon it becomes cyclic in any conforming implementation FSM with at most m states. The termination rule is formulated in terms of partial orders on the traces, defined by the reduction relation between the states. The improvement compared to [6] is the use of definitely reachable states instead of deterministically reachable states which shortens the traversal sets.

Given a specification FSM $\mathcal{S}$, a cover U_K of the set K of all definitely reachable states, states $s, s' \in K$, a preamble $\mathcal{P}_s$, and a non-empty trace $\beta \in Tr_{\mathcal{S}}(s)$, we define a (strict) partial order $\leq_{s'}$ on the set $T(\mathcal{P}_s)Pr(\beta) \cup U_K$, where $T(\mathcal{P}_s)Pr(\beta) = \{\alpha\gamma | \alpha \in T(\mathcal{P}_s)$ & $\gamma \in Pr(\beta)\}$, such that

1) for $\omega, \omega' \in \alpha Pr(\beta)$, $\alpha \in T(\mathcal{P}_s)$, $\omega \leq_{s'} \omega'$ if $|\omega| < |\omega'|$ and $\mathcal{S}$-after-$\omega \leq \mathcal{S}$-after-$\omega' \leq s'$; and
2) for $\omega \in U_K$, $\omega' \in T(\mathcal{P}_s)Pr(\beta)$, $\omega \leq_{s'} \omega'$ if $\omega \neq \omega'$ and $\mathcal{S}$-after-$\omega \leq \mathcal{S}$-after-$\omega' \leq s'$.

Let $C(T(\mathcal{P}_s)Pr(\beta), U_K, s')$ be a longest chain of the poset $(T(\mathcal{P}_s)Pr(\beta) \cup U_K, \leq_{s'})$, and $|C(T(\mathcal{P}_s)Pr(\beta), U_K, s')|$ be its length. Given a state $s \in K$, trace $\beta \in Tr_{\mathcal{S}}(s)$ is a *traversal* trace for the preamble $\mathcal{P}_s$ if $\Sigma_{s'\in R}|C(T(\mathcal{P}_s)Pr(\beta), U_K, s')| = m + 1$ for some set $R \in \mathcal{R}_{\mathcal{S}}$, where $\mathcal{R}_{\mathcal{S}}$ denotes the set of all maximal sets of pairwise *r*-distinguishable states of $\mathcal{S}$. For each traversal trace β we select one among such sets *R* in $\mathcal{R}_{\mathcal{S}}$ and denote it R_β.

The set of all possible traversal traces for the preamble $\mathcal{P}_s$ is a *traversal set* $N(U_K, \mathcal{P}_s)$. We illustrate the construction of traversal sets using our running example.

Example. We assume that any implementation machine has at most four states, i.e., $m = 4$. All the states of $\mathcal{S}$ are pairwise *r*-distinguishable, $\mathcal{R}_{\mathcal{S}} = \{\{1, 2, 3, 4\}\}$.

Consider state 1. The set of all completed traces of $\mathcal{P}_1$ contains just the empty word ε. To determine the traversal set for state 1, we start by considering all the traces of length one of this state and iteratively increasing their length until the above condition is satisfied. We have initially the traces *a*0, *a*1, *b*0, *c*0, *c*1 of state 1. For state $s = 1$ and trace $\beta = a0$, we construct the set $T(\mathcal{P}_s)Pr(\beta) \cup U_K$, where $T(\mathcal{P}_s)Pr(\beta) = \{a0\}$, $U_K = \{\varepsilon, a1, a0c1b1, c0, c1a1, c1, c0c1\}$, thus $T(\mathcal{P}_s)Pr(\beta) \cup U_K = \{\varepsilon, a0, a1, a0c1b1, c0, c1a1, c1, c0c1\}$. The longest chain $C(T(\mathcal{P}_s)Pr(\beta), U_K, 1)$ of the poset $(T(\mathcal{P}_s)Pr(\beta) \cup U_K, \leq_1)$, is $\{\varepsilon\}$, a longest chain $C(T(\mathcal{P}_s)Pr(\beta), U_K, 2)$ of the poset $(T(\mathcal{P}_s)Pr(\beta) \cup U_K, \leq_2)$ is a singleton, so is one for $C(T(\mathcal{P}_s)Pr(\beta), U_K, 4)$. A longest chain $C(T(\mathcal{P}_s)Pr(\beta), U_K, 3)$ is $\{a0, c0\}$, since $\mathcal{S}$-after-$a0$ = $\mathcal{S}$-after-$c0$ = 2, thus, $|C(T(\mathcal{P}_s)Pr(\beta), U_K, 3)| = 2$. For the set $R = \{1, 2, 3, 4\}$, we obtain $\Sigma_{s'\in R}|C(T(\mathcal{P}_s)Pr(\beta), U_K, s')| = 1 + 2 + 1 + 1 = 5$. Therefore, the trace *a*0 is a traversal trace for the preamble $\mathcal{P}_1$. Similarly, we conclude that the remaining traces of length one, *b*0, *c*0, *c*1 are also traversal traces for state 1. Thus, for the preamble $\mathcal{P}_1$, the traversal set $N(U_K, \mathcal{P}_1)$ becomes $\{a0, a1, b0, c0, c1\}$.

The traversal sets for the remaining preambles are constructed as above. $N(U_K, \mathcal{P}_2) = \{a0, b1, c1\}$, $N(U_K, \mathcal{P}_3) = \{a0, a1, b0, c1\}$ and $N(U_K, \mathcal{P}_4) = \{a1, b1, c1\}$.

Given a traversal trace $\beta \in N(U_K, \mathcal{P}_s)$, $s' \in R_\beta$, the set $Id(s', R_\beta)$ is used to identify that a given state cannot be a reduction of any state of the set R_β but state s'. In our example, we have that $Id(s, S) = Id(s, R_\beta)$ for each state *s*, since $R_\beta = \{1, 2, 3, 4\}$ for each trace β in the traversal sets obtained above. ♦

3.4 FSM Tests

We use FSMs to model tests from an implementation under test perspective: inputs (outputs) of the specification machine are also inputs (outputs) of a test. Thus, a tester, executing the test, applies inputs to an implementation FSM, observes its outputs and produces a corresponding verdict defined by a final state, *pass* or *fail*, reached by the test.

Definition 5. *An acyclic output-complete FSM* $\mathcal{U} = (U \cup \{pass, fail\}, u_0, I, O, h_{\mathcal{U}})$, *where pass and fail are designated deadlock states, is a* test *if* $(u, i, o, fail) \in h_{\mathcal{U}}$ *implies that* u-after-$io'\beta = pass$ *for some* $o' \neq o$ *and* $\beta \in Tr_{\mathcal{U}}(u\text{-after-}io')$.

Given a test $\mathcal{U}$, we further refer to traces which take $\mathcal{U}$ from the initial state to the *fail* state as *fail* traces and denote $Tr_{\mathcal{U}}^{\text{fail}}$ the set of all fail traces. *Pass* traces are defined as follows, $Tr_{\mathcal{U}}^{\text{pass}} = Tr_{\mathcal{U}_{u_0}} \setminus Tr_{\mathcal{U}}^{\text{fail}}$. Note that while fail traces are completed traces, pass traces can be proper prefixes of other traces. Test $\mathcal{U}$ is a *trivial* test if it is a trivial FSM with the initial *pass* state.

A test may have transitions with different inputs from a same state. In several work, this is not allowed in order to ensure the controllability of test execution. We leave the choice of inputs to the tester; assuming the following about the tester. If in a current state of the test, several inputs are defined, then executing such a test the tester simply selects one among alternative inputs during a particular test run. To execute another input defined in this state, the tester first uses a reset operation assumed to be available in any implementation to reset it to its initial state and then re-executes the preamble to this state. Test execution continues until no more unexecuted inputs in the test are left. Moreover, test execution is *adaptive*: depending on the observed output reaction to the previous input, the tester either chooses a next input to execute or just terminates the test run when an unexpected output is observed and it reaches the state *fail*.

To characterize conformance in this paper, we use the reduction relation and assume that the specification can be nondeterministic while implementations are deterministic, but both are complete machines. Given a complete FSM $\mathcal{S} = (S, s_0, I, O, h_\mathcal{S})$, let $\Im(\mathcal{S})$ be a set of complete deterministic (implementation) machines over the input alphabet I and the output alphabet O, called a *fault domain*. FSM $\mathcal{B} \in \Im(\mathcal{S})$ is a *conforming* implementation machine of $\mathcal{S}$ w.r.t. the reduction relation if $\mathcal{B} \leq \mathcal{S}$.

Definition 6. *Given the specification FSM $\mathcal{S}$, a test $\mathcal{U} = (U, u_0, I, O, h_\mathcal{U})$, and an implementation FSM $\mathcal{B} \in \Im(\mathcal{S})$,*

- $\mathcal{B}$ passes *the test $\mathcal{U}$, if the intersection $\mathcal{B} \cap \mathcal{U}$ has no state, where the test $\mathcal{U}$ is in the state fail. The test $\mathcal{U}$ is* sound *for FSM $\mathcal{S}$ in $\Im(\mathcal{S})$ w.r.t. the reduction relation, if any $\mathcal{B} \in \Im(\mathcal{S})$, which is a reduction of $\mathcal{S}$, passes the test $\mathcal{U}$.*
- $\mathcal{B}$ fails *$\mathcal{U}$ if the intersection $\mathcal{B} \cap \mathcal{U}$ has a state, where the test $\mathcal{U}$ is in the state fail. The test $\mathcal{U}$ is* exhaustive *for FSM $\mathcal{S}$ in $\Im(\mathcal{S})$ w.r.t. the reduction relation, if any $\mathcal{B} \in \Im(\mathcal{S})$, which is not a reduction of $\mathcal{S}$, fails the test $\mathcal{U}$.*
- *The test $\mathcal{U}$ is* complete *for FSM $\mathcal{S}$ in $\Im(\mathcal{S})$ w.r.t. the reduction relation, if it is sound and exhaustive in $\Im(\mathcal{S})$ w.r.t. the reduction relation.*

The set $\Im(\mathcal{S})$ that contains all complete deterministic FSMs with at most m states is denoted $\Im_m(\mathcal{S})$. A test is *m-complete* if it is complete in the fault domain $\Im_m(\mathcal{S})$.

4 Adaptive Testing

In this section, we propose an algorithm for adaptive testing of a complete deterministic implementation FSM $\mathcal{B}$ with at most m states; the algorithm yields the verdict pass if $\mathcal{B}$ is a reduction of a given specification FSM $\mathcal{S}$ and verdict fail if it is not a reduction of $\mathcal{S}$.

Test fragments defined in previous sections are used in the proposed method as in all the existing methods; namely, we reach states with preambles, apply traversal sets after reached states and check states with separators. However, preambles have to be executed in an adaptive way to reach states of the implementation FSM which match definitely reachable states of the specification machine (and only they, as opposed to methods for DFSMs), while the execution of separators is in general also adaptive. The key differences are related to traversal sets. First, since a conforming FSM may not implement all the traversal traces, we need to determine those present in a given implementation FSM. This is achieved by executing the input projections of the traversal traces. Second, learning these traces during test execution allows determining separators to check r-distinguishable states, traversed by them.

The algorithm includes also the construction of an FSM which represents all the traces observed during test execution. This machine is then used to show that an FSM which contains the observed traces for all implementation FSMs in $\Im_m(\mathcal{S})$ is an m-complete test for the FSM $\mathcal{S}$.

Algorithm 2 for adaptive testing of a deterministic implementation FSM

Input. Complete FSM $\mathcal{S}$, a set K of definitely reachable states, the set $Id(s, S)$ and preamble $\mathcal{P}_s$ for each $s \in K$, traversal sets $N(U_K, \mathcal{P}_s)$, sets $Id(s', R_\beta)$ for each $s' \in R_\beta$ and $\beta \in N(U_K, \mathcal{P}_s)$, and an implementation (black box) which behaves as a deterministic FSM $\mathcal{B}$.

Output. Verdict pass if $\mathcal{B}$ is a reduction of $\mathcal{S}$ or verdict fail if $\mathcal{B}$ is not a reduction of $\mathcal{S}$ and the FSM $\mathcal{G}_\mathcal{B}$ that contains all the observed traces of $\mathcal{B}$.

```
Initialize the two sets $T^{pass}$ and $T^{fail}$ of traces as the empty sets;
While there exists an unexecuted separator $\mathcal{R}(s, s')$ in the set $Id(s, S)$ for some $s \in K$
    Apply reset;
    Execute the preamble $\mathcal{P}_s$ until the observed trace is not in $\mathcal{P}_s$ or the designated state
    $s$ of the preamble is reached, let $\alpha$ be the observed trace;
      If the trace $\alpha$ is not in $\mathcal{P}_s$ add $\alpha$ to the set $T^{fail}$ and terminate with the verdict fail;
      Otherwise, let the observed completed trace of $\mathcal{P}_s$ be $\alpha_s$; add the trace $\alpha_s$ to $T^{pass}$;
      Mark "executed" all the separators in $Id(s, S)$ which have the same set of traces
      to the state $s$ as $\mathcal{R}(s, s')$ and execute the separator $\mathcal{R}(s, s')$, let $\beta$ be the observed
      trace;
          If the designated state $s$ of the separator is not reached add the trace $\alpha_s\beta$ to
          $T^{fail}$ and terminate with the verdict fail;
          Otherwise, add the trace $\alpha_s\beta$ to $T^{pass}$;
 End While;
 While there exists an unexplored state $s \in K$
    While there exists an unexecuted trace in the traversal set $N(U_K, \mathcal{P}_s)$, let $\gamma$ be the
    input projection of a longest unexecuted traversal trace
      Apply reset;
      Execute the preamble $\mathcal{P}_s$ (and observe the completed trace $\alpha_s$ of $\mathcal{P}_s$);
      Apply the inputs of $\gamma$ one by one until the observed trace is not in $Tr(\mathcal{S}/s)$ or the
      trace is in $N(U_K, \mathcal{P}_s)$; let $\beta$ be the observed trace with the input projection $\nu$;
```

```
                If the trace $\beta \notin Tr(\mathcal{S}/s)$ add the trace $\alpha_s\beta$ to $T^{\text{fail}}$ and terminate with the
                verdict fail;
                Otherwise, i.e., if $\beta \in N(U_K, \mathcal{P}_s)$ then mark "executed" each trace in $N(U_K, \mathcal{P}_s)$
                whose input projection has the prefix $\nu$;
                While for some prefix $\sigma$ of $\beta$ such that $\alpha_s\sigma \in C(T(\mathcal{P}_s)Pr(\beta), U_K, s')$, $s' \in R_\beta$,
                there exists an unexecuted separator $\mathcal{R}(s', s'')$ in the set $Id(s', R_\beta)$
                    Apply reset;
                    Execute the preamble $\mathcal{P}_s$ (and observe the completed trace $\alpha_s$ of $\mathcal{P}_s$);
                    Apply the input projection of $\sigma$, let $\eta$ be the observed trace;
                    Add to $T^{\text{pass}}$ the observed trace $\alpha_s\eta$;
                    Mark "executed" all the separators in $Id(s', R_\beta)$ which have the same set
                    of traces to the designated state $s'$ as the unexecuted separator $\mathcal{R}(s', s'')$
                    and execute the separator, let $\kappa$ be the observed trace;
                    If the designated state $s'$ of the separator is not reached then add the
                    trace $\alpha_s\eta\kappa$ to $T^{\text{fail}}$ and terminate with the verdict fail;
                    Otherwise, add the trace $\alpha_s\eta\kappa$ to $T^{\text{pass}}$;
                End While;
            End While;
        Mark the state $s$ "explored";
        End While;
        Terminate with the verdict pass;
    Derive an FSM $\mathcal{G}_\mathcal{B}$ with the set of traces $pref(T^{\text{pass}}) \cup T^{\text{fail}}$, such that each completed
    trace in $T^{\text{pass}}$ takes the FSM $\mathcal{G}_\mathcal{B}$ to the deadlock state pass and each trace in $T^{\text{fail}}$ takes $\mathcal{G}_\mathcal{B}$
    to the deadlock state fail. ♦
```

Theorem 1. *Given a deterministic FSM $\mathcal{B} = (B, b_0, I, O, h_\mathcal{B})$ with at most m states, let Algorithm 2 be used for the adaptive testing of FSM $\mathcal{B}$ against a given specification FSM $\mathcal{S}$. Then the verdict pass is produced if $\mathcal{B}$ is a reduction of $\mathcal{S}$ while the verdict fail is produced if $\mathcal{B}$ is not a reduction of $\mathcal{S}$.*

Before proving Theorem 1 we first illustrate Algorithm 2 with our running example.

Example. Assume we are given the deterministic implementation FSM $\mathcal{B}$ in Figure 1(b). For simplicity the FSM $\mathcal{B}$ has the state labels as in the FSM $\mathcal{S}$; it is then a submachine of that FSM, differing in the transitions (1, *a*, 0, 3) and (3, *a*, 1, 4) of $\mathcal{S}$ which are absent in the FSM $\mathcal{B}$.

We initialize the two sets T^{pass} and T^{fail} as the empty sets and consider state 1 in the set *K*. The preamble $\mathcal{P}_1$ is a trivial FSM, so we just add the trace ε to T^{pass}. $Id(1, S)$ includes the separators: $\mathcal{R}(1, 2)$, $\mathcal{R}(1, 3)$, and $\mathcal{R}(1, 4)$ which we execute by applying first input *b* to the implementation machine, observing the completed trace *b*0 and then, after reset, input *a* which results in output 1, followed by input *a* observing the completed trace *a*1*a*0. We add the traces *b*0 and *a*1*a*0 to T^{pass}. Next, we execute the preamble $\mathcal{P}_2$ and observe the completed trace *a*1 of this preamble, which we add to T^{pass}. Notice that from now on, executing the preamble $\mathcal{P}_2$ reduces to applying input a and observing trace $a1$. To execute the separators in $Id(2, S)$, we apply input a followed by b, observing the trace $a1b1$; finally a followed by a, observing the trace $a1a0$, which we add to T^{pass}. Executing the preamble $\mathcal{P}_3$, we obtain the trace $c1a1$ and add it to T^{pass}. After the subsequent execution of three separators, we add to T^{pass} the traces $c1a1a0b1$ and $c1a1b0$. Executing the preamble $\mathcal{P}_4$ we obtain the trace $c1$ and add it to T^{pass}. After the subsequent execution of the three separators, we add to T^{pass} the traces $c1a1$ and $c1b1$. Completing the execution of the preambles along with the separators, we obtain the following pass traces $T^{pass} = \{ \varepsilon, b0, a1a0, a1, a1b1, c1a1, c1a1a0b1, c1a1b0, c1b1\}$.

Next, we have to execute traversal sets after their corresponding preambles followed by separators. Consider state 1 and $N(U_K, \mathcal{P}_1) = \{a0, a1, b0, c0, c1\}$. We apply a and observe the trace $a1$; executing the separators in $Id(2, S)$ we observe the traces which are already in T^{pass}, namely, $a1b1$ and $a1a0$. We apply b and observe the trace $b0$; executing the separators in $Id(1, S)$ we observe the traces $b0b0$ and $b0a1a0$. Finally we apply c and observe the trace $c1$; executing the separators in $Id(4, S)$ we observe the traces which are already in T^{pass}, namely, $c1a1$ and $c1b1$.

Consider state 2 and $N(U_K, \mathcal{P}_2) = \{a0, b1, c1\}$. We obtain the following observations: $a1a0a0$, $a1a0b1$, $a1b1a0$, $a1b1b1$, $a1c1a1$, and $a1c1b1$. For state 3 and $N(U_K, \mathcal{P}_3) = \{a0, a1, b0, c1\}$, we have $c1a1a0a0$, $c1a1a0b1$, $c1a1b0a0b1$, $c1a1b0b0$, $c1a1c1a1$, $c1a1c1b1$. Finally, for state 4 and $N(U_K, \mathcal{P}_4) = \{a1, b1, c1\}$ we obtain the following observations: $c1a1a0b1$, $c1a1b0$, $c1b1a0$, $c1b1b1$, $c1c1a1$, $c1c1b1$.

The resulting set of completed pass traces becomes $\{a1a0a0, a1a0b1, a1b1a0, a1b1b1, a1c1a1, a1c1b1, b0a1a0, b0b0, c1a1a0a0, c1a1a0b1, c1a1b0a0b1, c1a1b0b0, c1a1c1a1, c1a1c1b1, c1b1a0, c1b1b1, c1c1a1, c1c1b1\}$. The set of fail traces T^{fail} is empty, since the FSM $\mathcal{B}$ is a reduction of the FSM $\mathcal{S}$. The algorithm terminates with the verdict pass. The obtained acyclic FSM $\mathcal{G}_\mathcal{B}$ has the set of traces $Pref(T^{pass})$ and each completed trace takes the FSM $\mathcal{G}_\mathcal{B}$ to the deadlock state *pass.* ♦

Compare now the obtained result with a preset test suite returned by the method from [6]. That method constructs traversal sets only for deterministically reachable states. As a result the traces in these traversal sets are longer since each deterministically reachable state is also definitely reachable, but the converse is not true. Since there are only two deterministically reachable state in the specification FSM in Figure 1, the test suite will contain all the input sequences of length three appended with corresponding separators, i.e., the total length of a test suite is more than $5{\cdot}3^3$. Here we notice that the method for constructing a complete test suite in [2] also uses only deterministically reachable states.

Now we return to the proof of the theorem.

Proof of Theorem 1. By construction, if $\mathcal{B}$ is a reduction of the specification FSM $\mathcal{S}$, then only the verdict pass can be produced by Algorithm 2. In fact, according to the algorithm, each observed trace is a trace of $\mathcal{S}$ and is a pass trace of the test, so the verdict fail cannot be produced.

Consider now FSM $\mathcal{B} \in \Im_m(\mathcal{S})$ that is not a reduction of $\mathcal{S}$ and for an observed completed trace α_s of the preamble $\mathcal{P}_s$, $s \in K$, and for each observed trace $\alpha_s\sigma \in C(T(\mathcal{P}_s)Pr(\beta), U_K, s')$, $\beta \in N(U_K, \mathcal{P}_s)$, $s' \in R_\beta$, no verdict fail was produced. We now show that in this case, there exist $s \in K$ and $\beta \in N(U_K, \mathcal{P}_s) \cap Tr(\mathcal{B}\text{-after-}\alpha_s)$ such that for the observed completed trace α_s of the preamble $\mathcal{P}_s$, there exists a set of sequences $M = \{\mu_1, ..., \mu_{m+1}\} \subseteq \cup_{s' \in R_\beta} C(T(\mathcal{P}_s)Pr(\beta), U_K, s')\}$ with the following property. The set $\{(\mathcal{S} \cap \mathcal{B})\text{-after-}\mu_j \mid j = 1, \ldots, m+1\}$ contains states (s', b) and (s'', b) such that $s' \le s_1$ and $s'' \le s_2$ for some states $s_1, s_2 \in R_\beta$.

Let Q' be the set of states that are reachable in the intersection $\mathcal{S} \cap \mathcal{B}$ via observed completed traces of all preambles in the cover U_K, i.e., $Q' = \{(\mathcal{S} \cap \mathcal{B})\text{-after-}\alpha \mid \alpha \in U_K\}$. Let also ν be a shortest trace from a state of the set Q' to a state $q = sb$ such that some input i is not defined in state q under i (Corollary 1), i.e., the output of $\mathcal{B}$ at state b under input i is not in the set of outputs at state s of the specification FSM $\mathcal{S}$. The property of ν being a shortest such trace means that for each trace $\rho\delta \in Tr_{\mathcal{S}}$, where ρ is a completed trace of a preamble $\mathcal{P}_{s'}$, $s' \in K$, and δ possessing the same property, i.e., such that the state $(\mathcal{S} \cap \mathcal{B})\text{-after-}\rho\delta$ has an undefined input, it holds that $|\delta| \ge |\nu|$. Since $\mathcal{B}$ is not a reduction of $\mathcal{S}$, the intersection $\mathcal{S} \cap \mathcal{B}$ is initially connected, and the set K has the initial state, such a trace ν exists. By definition of traversal sets $N(U_K, \mathcal{P}_s)$ and since no verdict fail was produced for observed traces of FSM $\mathcal{B}$, there exist a preamble $\mathcal{P}_s$ and an observed trace $\alpha_s\beta$, where $\alpha_s \in T(\mathcal{P}_s)$, and β is a prefix of ν, with the property that there exists $R_\beta \in \mathsf{R}_{\mathcal{S}}$, such that $\mathcal{S}\text{-after-}\alpha_s\beta \in R_\beta$ and $\Sigma_{s' \in R_\beta}|C(T(\mathcal{P}_s)Pr(\beta), U_K, s')| = m + 1$.

For each state $s' \in R_\beta$, consider a longest chain of the poset $(\alpha_s Pr(\beta) \cup U_K, \le_{s'})$; let $M = \{\mu_1, \ldots, \mu_k, \mu_{k+1}, ..., \mu_{m+1}\}$. Without loss of generality, we assume that $\mu_1, \ldots, \mu_k$ are the observed completed traces of the set U_K while $\mu_{k+1}, ..., \mu_{m+1}$ are sequences of the set $\alpha_s Pr(\beta)$.

Consider the corresponding $m + 1$ states of FSM $\mathcal{B}$, $\mathcal{B}\text{-after-}\mu_1, \ldots, \mathcal{B}\text{-after-}\mu_{m+1}$. Since $\mathcal{B}$ has at most m states there exist $1 \le j < r \le m + 1$ such that $\mathcal{B}\text{-after-}\mu_j = \mathcal{B}\text{-after-}\mu_r$. By the definition of the poset, either $\mathcal{S}\text{-after-}\mu_j \le \mathcal{S}\text{-after-}\mu_r$ or $\mathcal{S}\text{-after-}\mu_j \not\simeq \mathcal{S}\text{-after-}\mu_r$. Let $\mathcal{S}\text{-after-}\mu_j \le \mathcal{S}\text{-after-}\mu_r$. By definition of the poset $(\alpha_s Pr(\beta) \cup U_K, \le_{s'})$, two cases are possible, either $\mu_j \in U_K$ and $\mu_r \in \alpha_s Pr(\beta)$ or $\mu_j, \mu_r \in \alpha_s Pr(\beta)$.

Consider the case when $\mu_j \in U_K$ and $\mu_r \in \alpha_s Pr(\beta)$. As Algorithm 2 produces the verdict pass when the implementation $\mathcal{B}$ is tested, the following holds. The trace β' obtained from $\alpha_s\beta$ by deleting the prefix μ_r, is a trace of FSM $\mathcal{B}$ at state $\mathcal{B}\text{-after-}\mu_r = \mathcal{B}\text{-after-}\mu_j$. If β' is not a trace of FSM $\mathcal{S}$ at state $\mathcal{S}\text{-after-}\mu_j$ then a proper prefix β'' of the trace β' takes the FSM $\mathcal{S} \cap \mathcal{B}$ from state $(\mathcal{S}\text{-after-}\mu_j, \mathcal{B}\text{-after-}\mu_j)$ to a state $(\mathcal{S}\text{-after-}\mu_j\beta'', \mathcal{B}\text{-after-}\mu_j\beta'')$ such that the state $(\mathcal{S}\text{-after-}\mu_j\beta'', \mathcal{B}\text{-after-}\mu_j\beta'')$ of the intersection $\mathcal{S} \cap \mathcal{B}$ has an undefined input. If β' is a trace of FSM $\mathcal{S}$ at state $\mathcal{S}\text{-after-}\mu_j$ then β' is a trace of $\mathcal{S} \cap \mathcal{B}$ at state $(\mathcal{S} \cap \mathcal{B})\text{-after-}\mu_j$ and thus, a shorter trace ν_r obtained from ν by deleting the prefix μ_r is a trace of FSM $\mathcal{S} \cap \mathcal{B}$ to a state ($\mathcal{S}$-after-$\mu_j\nu_r$, $\mathcal{B}$-after-$\mu_j\nu_r$) with an undefined input.

Consider now the case when μ_j, $\mu_r \in \alpha_s Pr(\beta)$. Similar to the previous case, the trace β could also be shortened by deleting the part between two states $\mathcal{S}$-after-μ_j and $\mathcal{S}$-after-μ_r, as $\mathcal{S}$-after-$\mu_j \leq \mathcal{S}$-after-μ_r.

Both cases contradict the fact that the trace ν that contains β as a prefix is a shortest trace from a state of the set Q' to a state $q = sb$ with an undefined input i.

Thus, $\mathcal{S}$-after-$\mu_j \not\simeq \mathcal{S}$-after-$\mu_r$, i.e., the set $\{(\mathcal{S} \cap \mathcal{B})$-after-$\gamma \mid \gamma \in M\}$ contains states (s, b) and (s', b) such that $s \leq s_1$ and $s' \leq s_2$ for some states $s_1, s_2 \in R_\beta$, thus $s \not\simeq s'$. According to Algorithm 2, for the observed completed trace α_s of the preamble $\mathcal{P}_s$, $s \in K$, after each $\gamma \in C(T(\mathcal{P}_s)Pr(\beta), U_K, s')$, $s' \in R_\beta$, there will be an identifier $Id(s', R_\beta)$ executed. Correspondingly, an identifier $Id(\mathcal{S}$-after-$\mu_j, R_\beta)$ after the trace μ_j and an identifier $Id(\mathcal{S}$-after-$\mu_r, R_\beta)$ after the trace μ_r will be executed. Since $\mathcal{S}$-after-$\mu_j \not\simeq \mathcal{S}$-after-$\mu_r$, $\mathcal{B}$ is deterministic and $\mathcal{B}$-after-$\mu_j = \mathcal{B}$-after-μ_r, at least for one of the separators the designated state would not be reached, i.e., the verdict fail will be produced. ♦

Finally, we demonstrate how the obtained result is related to the problem of m-complete test generation. Suppose that all the deterministic FSMs with at most m states in the fault domain $\Im_m(\mathcal{S})$ can be explicitly enumerated and for each such FSM $\mathcal{B}$, an FSM $\mathcal{G}_\mathcal{B}$ is derived by Algorithm 2. Let the FSM $\mathcal{U}$ be the union of all FSMs $\mathcal{G}_\mathcal{B}$ which, if it is not output-complete, is completed as follows. For each trace αio of $\mathcal{U}$ if there exists $o' \in O$ such that $\mathcal{U}$ has no trace $\alpha io'$ then a completed pass trace $\alpha io'$ is added if $\alpha io'$ is a trace of the specification FSM $\mathcal{S}$; otherwise, a fail trace $\alpha io'$ is added. The reason is that such a trace $\alpha io'$ does not belong to any machine in $\Im_m(\mathcal{S})$; it has to be added since a test is by definition output-complete. In fact, a test can be output-completed in arbitrary way, since those traces will never be observed in testing deterministic FSMs with at most m states.

Theorem 2. *Given an FSM $\mathcal{S}$ and a test $\mathcal{U}$ derived by the above procedure, the test $\mathcal{U}$ is m-complete for $\mathcal{S}$, i.e., it is complete in the fault domain $\Im_m(\mathcal{S})$ of complete deterministic FSMs.*

Proof. A trace of test $\mathcal{U}$ is a pass trace if and only if this trace is a trace of the specification FSM $\mathcal{S}$. Consider FSM $\mathcal{B} \in \Im_m(\mathcal{S})$ that is not a reduction of $\mathcal{S}$; by construction, an FSM $\mathcal{G}_\mathcal{B}$ derived by the above algorithm has a fail trace that is a trace of $\mathcal{B}$ and thus, test $\mathcal{U}$ has a fail trace that is a trace of $\mathcal{B}$. ♦

Such a characterization of the relationship between the executed test and m-complete tests is of a more theoretical than practical interest, as in testing one usually deals with a given implementation and not with an arbitrary collection of them.

5 Related Work

Most of the previous work in test generation from NFSMs relies on the execution of all test fragments forming the complete tests, see e.g., [4, 6]. The proposed method requires adaptive execution avoiding tests which are not related to a given implementation. It also differs in the way test fragments are treated. The construction of traversal sets follows the most recent idea elaborated for deriving preset complete tests in our previous work [6], which improves previous proposals, (see [6] for more references) including those used for adaptive testing such as [2, 3, 7]. In all the previous work, the test fragment addressing state reachability includes transfer sequences only to deterministically reachable states, which is extended in this paper by considering definitely reachable states.

The notions of preamble used in this paper and finite transfer tree considered in [11] serve the same purpose of modeling an adaptive process of transferring an NFSM into a desired state. Differently from that work, we use, instead of a tree, a state machine. Modeling preamble as NFSM allows us not only to establish a direct relation with a specification machine, namely, that a preamble is just a certain submachine of the specification FSM, moreover if it exists then any of its reductions possesses the preamble. This allows us to solve its existence and construction problems in a simple intuitive way.

The notion of separator is similar to that of state distinguishing strategies considered in [1] and in [11]. Differently from that work, we not only use an FSM to describe a strategy, but also offer an exact characterization of all state distinguishing strategies in the form of an FSM, called a canonical separator, which is obtained from the self-product of the specification NFSM. A distinguishing strategy becomes then a certain submachine of the canonical separator. The method of [1] allows to test whether a given FSM is *r*-compatible with the specification FSM (but not that the former is a reduction of the latter, as we do), and [3] extends this result to a set of FSMs. The work in [5] also addresses adaptive testing of nondeterministic machines, but its results cannot be used to prove that a given DFSM is a reduction of the specification machine. Compared to [7], we pre-compute all the test fragments, simplifying the test generation process.

6 Conclusion

In this paper, we addressed the problem of adaptive testing of a deterministic implementation machine from its nondeterministic specification. We proposed a method for defining test fragments, combining and executing them in adaptive manner against a given implementation FSM such that the test execution terminates with the verdict pass if and only if it is a reduction of the specification. The novelty of the method lies in the use of definitely reachable states missed by the previous methods and selective execution of test fragments depending on observed traces in adaptive test execution.

Our current work concerns the generalization of the obtained results to nondeterministic implementations. As a future work, it would be interesting to investigate a combination of the proposed method with the specification refinement approach.

Acknowledgement. The authors want to thank a reviewer for useful comments.

References

1. R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing Tests for Nondeterministic and Probabilistic Machines. 27th ACM Symp. on Theory of Comp., 1995, pp. 363-372.
2. R. M. Hierons. Testing from a Non-Deterministic Finite State Machine Using Adaptive State Counting, IEEE Transactions on Computers, 2004, 53(10), pp. 1330-1342.
3. M. L. Gromov, N. V. Evtushenko and A. V. Kolomeets. On the Synthesis of Adaptive Tests for Nondeterministic Finite State Machines, Progr. and Comp. Software, 2008, 34(6), pp. 322-329.
4. G. L. Luo, G. v. Bochmann, and A. Petrenko. Test Selection Based on Communicating Nondeterministic Finite-State Machines Using a Generalized Wp-method. IEEE Transactions on Software Engineering, 1994, 20(2), pp. 149–161.
5. L. Nachmanson, M. Veanes, W. Schulte, N. Tillmann, W. Grieskamp. Optimal Strategies for Testing Nondeterministic Systems, ISSTA 2004, Software Eng. Notes, ACM, 29, pp. 55–64.
6. A. Petrenko, N. Yevtushenko. Conformance Tests as Checking Experiments for Partial Nondeterministic FSM, Proceedings of the 5th International Workshop on Formal Approaches to Testing of Software (FATES 2005), 2005, LNCS 3997, pp. 118-133.
7. A. Petrenko, N. Yevtushenko. Refining Specifications in Adaptive Testing of Nondeterministic Finite State Machines, Vestnik Tomskogo gos. universiteta, 2009, 1(6), pp. 99-114.
8. A. Petrenko, N. Yevtushenko, and G. v. Bochmann. Testing Deterministic Implementations from their Nondeterministic Specifications, Proceedings of the IFIP Ninth International Workshop on Testing of Communicating Systems, 1996, pp. 125-140.
9. A. Petrenko and N. Yevtushenko. Testing from Partial Deterministic FSM Specifications, IEEE Transactions on Computers, 2005, 54(9), pp.1154-1165.
10. A. Simao, A. Petrenko, and N. Yevtushenko. Generating Reduced Tests for FSMs with Extra States, Proceedings of TestCom/Fates, 2009, LNCS 5826, pp. 129–145.
11. F. Zhang and T. Cheung. Optimal Transfer Trees and Distinguishing Trees for Testing Observable Nondeterministic Finite-State Machines, IEEE Transactions on Software Engineering, 2003, 29(1), pp. 1-14.