On-the-Fly Construction of Adaptive Checking Sequences for Testing Deterministic Implementations of Nondeterministic Specifications

Nina Yevtushenko[1], Khaled El-Fakih[2(✉)], and Anton Ermakov[1]

[1] Tomsk State University, Tomsk, Russia
nyevtush@gmail.com, antonermak@inbox.ru
[2] American University of Sharjah, Sharjah, UAE
kelfakih@aus.edu

Abstract. A method is proposed for deriving an adaptive checking sequence for a given deterministic implementation of a nondeterministic Finite State Machine (FSM) specification with respect to the reduction relation. The implementation is non-initialized, i.e., there is no reliable reset input. In order to obtain a sequence of reasonable length, in the proposed technique, we consider specifications with adaptive distinguishing test cases and adaptive transfer sequences. In fact, we show how under these considerations we can on-the-fly derive a checking sequence where the head part establishes the one-to-one correspondence between states of the implementation and the specification and if established the second part of the sequence is constructed for checking the one-to-one correspondence between transitions of the implementation and a submachine of the specification FSM. The latter construction appropriately utilizes information from the first part to reach and check intended transitions.

Keywords: Nondeterministic finite state machines · Reduction relation · Fault model · Test derivation · Distinguishing test case · Definitely reachable states · Adaptive sequence

1 Introduction

Finite State Machine (FSM) based test derivation is widely used when deriving conformance tests in many application domains such as sequential circuits, communication protocols, web-services, etc. There are many approaches for FSM-based test derivation that are summarized in many surveys such as [2, 4, 20, 24]. In many approaches, such as in the W-method [3] and its many derivatives, both the specification and implementation FSMs are assumed to be initialized and thus, tests (input sequences or traces) are derived from a given initialized specification FSM; these tests are concatenated by a reliable reset that brings the machine to the initial state. Many other approaches do not rely on the existence of such possibly expensive resets and derive so-called checking sequences consisting of one test without resets. The reader may refer to [7–13] for some approaches and summary of existing work on deriving and reducing length of

© IFIP International Federation for Information Processing 2016
Published by Springer International Publishing AG 2016. All Rights Reserved

checking sequences for deterministic FSMs. In general, while constructing a checking sequence for deterministic FSMs, as there are no resets, one may rely on the so-called synchronizing (or homing) sequence that takes the FSM from any state to the known state in addition to an input sequence that can distinguish two different states of the machine.

Nowadays, design and analysis of non-deterministic systems is capturing a lot of attention. Nondeterminism can occur due to several reasons, such as limited observability, abstraction, etc. Accordingly, in this paper, we consider the derivation of checking sequences for observable nondeterministic FSMs. A nondeterministic machine is *observable* if for each state and input the machine can have many outgoing transitions under the input as long as different outputs are produced at these transitions. Otherwise, the machine is *non-observable*. We consider observable machines as it is known that any non-observable specification machine can be transformed into an observable one with the same behavior.

Petrenko *et al.* [14] proposed a method for deriving a checking sequence for a complete nondeterministic FSM with respect to the equivalence relation under appropriate assumptions about the specification FSM and the fault domain. The specification FSM has to have a distinguishing input sequence for which the sets of output responses at any two different states do not coincide. Since an implementation under test (IUT) can be nondeterministic, the authors also rely on the all-weather conditions assumption. In [21] the authors extended the work considering the derivation of a checking sequence with respect to the reduction relation. Resetting is still used yet only in one phase of the construction approach. Ermakov [6] presented a method for deriving an adaptive checking sequence with respect to the reduction relation under the assumption that the specification has a separating sequence, i.e., an input sequence for which the sets of output responses at any two different states are disjoint. The specification FSM has also to be deterministically connected, i.e., each state is deterministically reachable from any other state while an IUT is a complete deterministic FSM. A checking sequence is *adaptive* if the selection of the next input to be applied to an IUT depends on the outputs produced by the IUT to previously applied inputs. As in the other above approaches, the approach given in [6] also uses resetting. In this paper, we reduce the limitation considered in the above papers about the use of resets. Moreover, differently from [6] we show how to effectively use adaptive transfer and distinguishing sequences when deriving an adaptive checking sequence as such adaptive sequences can exist when there are no preset ones; in addition, such adaptive sequences can be shorter [1, 17, 19]. More precisely, we construct an adaptive checking sequence from a given non-deterministic observable FSM against a given complete deterministic IUT assuming that the specification FSM has adaptive transfer sequences as well as an adaptive distinguishing sequence (a distinguishing test case) of reasonable length. The existence of an adaptive transfer sequences means that every state of the machine is definitely reachable from any other state. We show that in this case, when testing with respect to the reduction relation, each state of the specification FSM is required to be implemented in an IUT. As usual, we also assume that the behavior of the IUT is not known, we only know that the number of states of the IUT does not exceed that of the specification. Under the above assumptions, an IUT is a reduction of the specification machine if and only if the IUT is isomorphic to a complete submachine of the specification FSM and thus, when testing it is enough to establish the one-to-one correspondence between states and transitions of the IUT and states and transitions of an appropriate submachine in the specification FSM. In other words, each transition of the IUT has to be traversed and an adaptive distinguishing sequence has to be applied for verifying the final state of the transition. This approach allows us to derive checking sequences of reasonable length when an adaptive distinguishing sequence has polynomial length with respect to the number of states of the specification FSM.

This paper is organized as followed. Section 2 includes preliminaries with related definitions. Section 3 includes the considered fault model and Sect. 4 includes the checking sequence construction method with related propositions and a simple application example. Section 5 concludes the paper.

2 Preliminaries

A *finite state machine* (FSM), or simply a *machine*, is a 4-tuple S = $\langle S, I, O, h_S \rangle$, where S is a finite nonempty set of states, I and O are finite input and output alphabets, and $h_S \subseteq S \times I \times O \times S$ is a (*behavior*) *transition relation*. FSM S is *nondeterministic* if for some pair $(s, i) \in S \times I$ there can exist several pairs $(o, s') \in O \times S$ such that $(s, i, o, s') \in h_S$. FSM S is *complete* if for each pair $(s, i) \in S \times I$ there exists $(o, s') \in O \times S$ such that $(s, i, o, s') \in h_S$. FSM S is *observable* if for each two transitions (s, i, o, s_1), $(s, i, o, s_2) \in h_S$ it holds that $s_1 = s_2$. FSM S is *initialized* if it has the designated initial state s_1, written S/s_1. Thus, an initialized FSM is a 5-tuple $\langle S, I, O, h, s_1 \rangle$. In the following, we consider observable and complete FSMs if the contrary is not explicitly stated.

A *trace* of S at state s is a sequence of input/output pairs of consecutive transitions starting from state s. Given a trace $i_1o_1 \ldots i_ko_k$ at state s, the input projection $i_1 \ldots i_k$ of the trace is a *defined* input sequence at state s. For an observable nondeterministic FSM, if $\gamma = i_1o_1 \ldots i_ko_k$ is a trace at a state s, then there exists a unique sequence of consecutive transitions $(s, i_1, o_1, s_1)(s_1, i_2, o_2, s_2)\ldots(s_{k\text{-}1}, i_k, o_k, s_k)$. As usual, for state s and a sequence $\gamma \in (IO)^*$ of input/output pairs, the γ-*successor* of state s is the set of all states that are reached from s by trace γ. If γ is not a trace at state s then the γ-successor of state s is empty or we simply say that the γ-successor of state s does not exist. For an observable FSM S, for any string $\gamma \in (IO)^*$, the cardinality of the γ-successor of state s is at most one. Given a subset S' of states, the γ-*successor* of S' is the union of γ-successors over all states of the set S'.

FSM S is *single-input* if at each state there is at most one defined input at the state, i.e., for each two transitions (s, i_1, o_1, s_1), $(s, i_2, o_2, s_2) \in h_S$ it holds that $i_1 = i_2$, and S is *output-complete* if for each pair $(s, i) \in S \times I$ such that the input i is defined at state s, there exists a transition from s with i for every output in O. An initialized FSM S is *acyclic* if the FSM transition diagram has no cycles. An initialized FSM S is (*initially*) *connected* if each state is reachable from the initial state. Given an input alphabet I and an output alphabet O, a *test case* TC(I, O) is an initially connected single-input output-complete observable initialized FSM T = (T, I, O, h_T, t_1) with an acyclic transition graph [22]. Given a complete FSM S over alphabets I and O, a test case TC(I, O) represents an adaptive experiment with the FSM S [15].

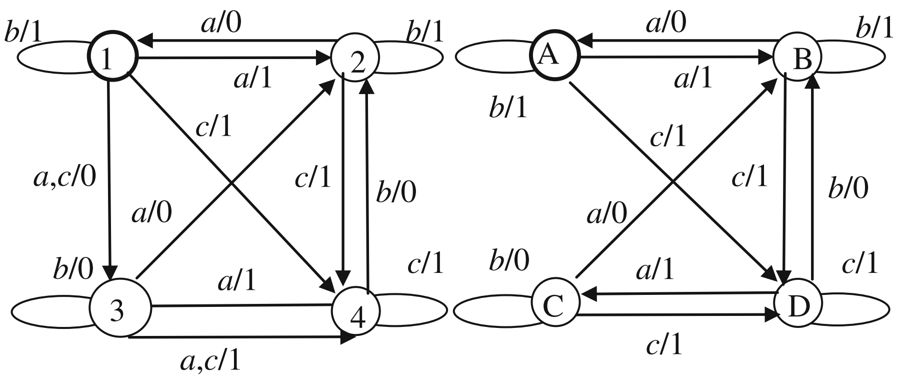

Fig. 1. **(a)** Specification FSM S **(b)** Implementation FSM P

If | I | > 1 then a test case is a partial FSM. A state $t \in T$ is a *deadlock* state of the FSM T if there are no defined inputs at this state. In general, given a test case T, the *length* (*height*) of the test case T is defined as the length of a longest trace from the initial state to a deadlock state of T and it specifies the length of the longest input sequence that can be applied to an FSM S during the experiment. A trace from the initial state to a deadlock state is a *complete* trace of a test case [23]. As usual, for complexity reasons, one is interested in deriving a test case with minimal length. A test case T is a *distinguishing* test case (DTC) for an FSM S if for every trace γ of T from the initial state to a deadlock state, γ is trace at most at one state of S. Sometimes, a distinguishing test case is called an *adaptive distinguishing sequence*.

Consider FSM S in Fig. 1a. Using the approach proposed in [18] a (adaptive) distinguishing test case can be constructed for FSM S (Fig. 2).

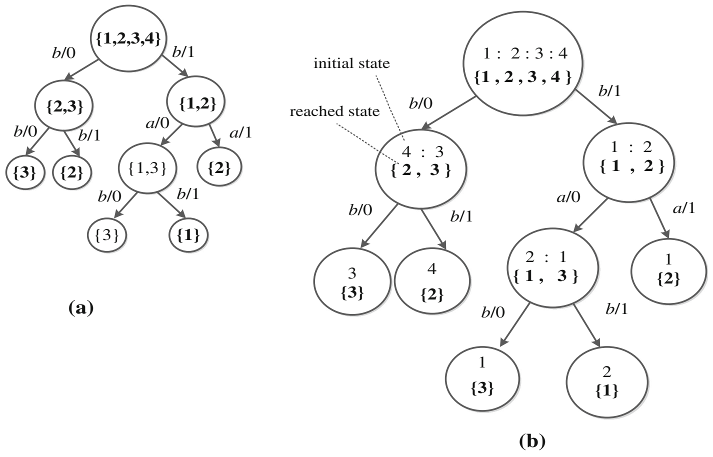

Fig. 2. **(a)** The distinguishing test case **(b)** The test case for (a) where source states are indicated

Given a complete observable FSM S = (S, I, O, h_S), state $s' \in S$ is *definitely-reachable* (*def-reachable*) from state $s \in S$ if there exists a test case $T_{s,s'}$ over I and O, initialized with the singleton $\{s\}$, and for every trace γ of $T_{s,s'}$ from the initial state to a deadlock state, the γ-successor of state s is either the empty set or $\{s'\}$.

We hereafter refer to such a test case $T_{s,s'}$ as a *def-transfer* test case from state s to state s' or as an *adaptive def-transfer sequence*.

In fact, a *def-transfer* test case is defined in [23] as an extension of a deterministic (d-)transfer sequence for states s and s'. All the traces of $T_{s,s'}$ take the FSM S from state s to state s'. When testing with respect to the reduction relation not each state of the specification FSM, except for the initial state, is required to be implemented in an implementation FSM P. However, if there exists a *def-transfer* test case $T_{s,s'}$ and state s is implemented in the reduction P of S then according to [23], state s' must be implemented in P.

In [23], necessary and sufficient conditions were established to check if state $s' \in S$ is definitely reachable from the initial state of the initialized FSM. Accordingly, when checking whether state s' is definitely reachable form state s, the initialized FSM S/s can be considered. Moreover, in [23] it is shown how a *def-transfer* test case $T_{s,s'}$ can be derived such that the length of $T_{s,s'}$ (if it exists) does not exceed the number of states of FSM S.

By direct inspection, one can assure that for every two different states s, s' of FSM S (Fig. 1a) there exists a *def*-transfer test case $T_{s,\ s'}$. As an example, consider $T_{1,2}$ in Fig. 3. If an IUT replies with 1 to the applied input a at state 1 then we know that the next state of the specification is state 2. If the output 0 is produced by the IUT then the specification reaches state 3 and we apply the input a again. If the IUT replies with 0 to the applied input a then we know that the next state of the specification is state 2. If the output 1 is produced by the IUT then the specification reaches state 4 and we apply an input b in order to reach state 2.

Consider the FSM in Fig. 1a. For states 2 and 1, 3 and 1, and 4 and 1, there exist deterministic transfer sequences, namely, state 1 is d-reachable from 2 by input sequence a, state 3 is d-reachable from 1 by input sequence $c\ b\ a$ and state 4 is d-reachable from 1 by input sequence $b\ a$. State 2 is d-reachable from 3 and 4 by input sequences $c\ b$ and b correspondingly, state 3 is d-reachable from 2 and 4 by input sequences $b\ a$ and a, while state 4 is d-reachable from 2 and 3 by input sequence b. By direct inspection, one can assure that $T_{1,3}$ and $T_{1,4}$ can be easily derived from the machine in Fig. 3 as states 3 and 4 are deterministically reachable from state 2.

In order to check if there exists a distinguishing test case for the specification FSM S, we can use the procedures proposed in [16, 17]. If a general procedure is used then the complexity can become exponential w.r.t. the number of FSM states [5]. The complexity of the procedure proposed in [16] is polynomial but it can be applied only for so-called merging free FSMs. A complete observable FSM is *merging free* if for each two different states s_1 and s_2, every input i and every output o, the non-empty i/o successors of s_1 and s_2 do not coincide. For a merging free FSM, a distinguishing test case exists if and only if a distinguishing test case exists for each pair of different

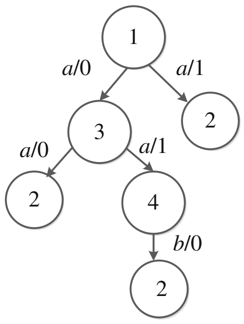

Fig. 3. A *def*-transfer test case for states 1 and 2 of the FSM in Fig. 1a

state of the FSM; the latter can be checked in polynomial time and the length of such test case is at most $n(n - 1)/2$ if the FSM has n states. Then a distinguishing test case for the FSM is derived step by step starting from a single pair and adding a new state for the set of initial states at each step. In [16], it is shown that the length of such test case is $O(n^3)$. The class of merging-free FSMs is big enough; at least it contains many deterministic FSMs which are used in practical applications [9].

Hereafter we use S to denote a complete observable nondeterministic specification machine while P denotes a complete deterministic IUT.

Given complete FSMs S and P, state p of the FSM P is a *reduction* of state s of the FSM S, written $p \le s$, if the set of traces of P at state p is a subset of that of S at state s; otherwise, p is not a reduction of state s, written $p \not\le s$. FSM P is a *reduction* of FSM S if for each state p there exists state s such that $p \le s$.

Given complete FSM S, two different states s_1 and s_2 and an input sequence α, α is a *separating* sequence of states s_1 and s_2 if the sets $out(s_1, \alpha)$ and $out(s_2, \alpha)$ are disjoint. If α separates each pair of different states then α is a separating sequence for FSM S. For non-deterministic observable machines the tight upper bound on the length of a separating sequence of two states is known to be exponential with respect to number of FSM states [25] while for deterministic FSMs the length of an adaptive distinguishing sequence (a distinguishing test case) is polynomial [26]. Moreover, if there exists a separating sequence then there exists a distinguishing test case but the opposite is not necessarily true. When an implementation is deterministic then the observation of n different replies to a separating sequence immediately means that the IUT has at least n states. For recognizing states of an implementation, a separating sequence sometimes can be replaced by state identifiers. An input sequence α is a *state identifier* of state s of FSM S if α is a separating sequence for each pair (s, s'), $s' \neq s$. As an example, the $b\ b$ is a state identifier for states 3 and 4 of the FSM in Fig. 1a and $b\ a\ b$ is a state identifier for states 1 and 2.

Given a test case TC of a complete observable specification S, a trace that takes TC from the initial state to a deadlock state is a *complete* trace; the set *CompleteTraces*(TC) is the set of all complete traces of TC. If there exists a distinguishing test case then any two different states s_1 and s_2 of FSM S are *r*-distinguishable and thus, any state of any complete FSM over alphabets *I* and *O* is not a reduction of two different states of FSM S [22]. The set of input projections of all complete traces of a distinguishing test case sometimes is called a *distinguishing set*. In [22], it is shown, that given a state *p* of a complete FSM over alphabets *I* and *O*, for any two states s_1 and s_2 of FSM S there always exists an input sequence of the distinguishing set such that the set of output responses of P at state *p* is not a subset of that at both states s_1 and s_2. Moreover, given a reduction P of FSM S, not each state of S has to be implemented in P. However, according to Proposition 1, if S has both a distinguishing test case and a *def*-transfer test case for each pair of different states then each state of S has to be implemented in P and there is the one-to-one correspondence between states and transitions of P and the corresponding states and transitions of an appropriate submachine of S. The latter allows the construction of shorter tests and reduces the efforts of checking if a given FSM P is a reduction of such specification FSM.

Two FSMs over the same input and output alphabets are *isomorphic* if there exists one-to-one correspondence between their states and transitions, i.e., if there exists one-to-one mapping $f: S \rightarrow P$ such that for any input i and any state s the 4-tuple $(s, i, o, s') \in h_{\mathrm{s}}$ if and only if $(f(s), i, o, f(s')) \in h_{\mathrm{P}}$.

Proposition 1. Given a complete observable FSM S with *n* states, let S have a distinguishing test case and each pair of different states *s* and *s'* of S have a *def*-transfer test case $\mathrm{T}_{s,\,s'}$. A complete observable FSM P that has at most *n* states is a reduction of S if and only if P is isomorphic to a submachine of S.

Proof. If P is isomorphic to a submachine of S then P is a reduction of S.

Let now P be a reduction of S, i.e., for each state *p* of P there exists state *s* of S such that $p \le s$. When there exists a distinguishing test case *DTC* for S, states of the FSM S are *r*-distinguishable and thus, any state of any complete FSM over alphabets *I* and *O* is not a reduction of two different states of FSM S [23]. Moreover, if $p \le s$ then a distinguishing test case *DTC* has a trace at state *p* of P that is not a trace at any other state of S. On the other hand, let state *s* of FSM S be implemented in P as state $p \le s$. Any state *s'* is *def*-reachable from *s* and thus also is implemented in P as $p' \le s'$. Therefore, each state of S is implemented in P and since P has at most *n* states, each state of S is implemented as a unique P state. Correspondingly, we can establish one-to-one correspondence $F_{DTC}: S \rightarrow P$ between states of FSMs S and P according to the given distinguishing test case.

Moreover, since P is a reduction of S the following holds. If there is a transition $p - i/o - p'$ where $p = F_{DTC}(s)$ and $p' = F_{DTC}(s')$ then S has a transition $s - i/o - s'$. □

Let S have a distinguishing test case *DTC*, each pair of different states *s* and *s'* of S have a *def*-transfer test case $\mathrm{T}_{s,\,s'}$ and the number of states of FSM P does not exceed that of S. FSM P is *DTC-compatible* with S if there exists one-to-one correspondence $F: S \rightarrow P$ such that for each state $s \in S$ it holds that the intersection of $Tr(\mathrm{S}/s) \cap Tr(\mathrm{P}/p) \cap CompleteTraces(DTC)$ is not empty if and only if $p = F(s)$.

Proposition 2. Given the specification FSM S that has a distinguishing test case *DTC*, let a deterministic complete FSM P be *DTC*-compatible with S, each pair of different states s and s' of S have a *def*-transfer test case $T_{s,\ s'}$ and the number of states of FSM P does not exceed that of S. For each state p of P, the distinguishing test case *DTC* has a complete trace α/β that is a trace at state p; moreover, α is a state identifier of state p in P.

Proof. In fact, if there exists one-to-one correspondence between states of S and P according to the distinguishing test case *DTC*, then for each two states s and s', $s' \neq s$, there exists a prefix of an input sequence of some complete trace α/β of *DTC* such that output responses at corresponding states $p = F(s)$ and $p' = F(s')$ are different. As *DTC* is a distinguishing test case of S and FSM P is complete and deterministic, the latter means that a corresponding input projection of trace α/β is a state identifier of state p. □

3 Fault Model for Deriving an Adaptive Checking Sequence

In FSM-based testing, it is assumed that the specification FSM describes the reference behavior while the fault domain FD contains each possible implementation FSM of the specification. In our case, the specification FSM S is complete and observable, moreover, S has a distinguishing test case and there exists a *def*-transfer test case for each pair of different states of S. The conformance relation is the reduction relation while any IUT of the FD is complete and deterministic and the number of its states does not exceed that of the specification FSM. In other words, we implicitly assume that the nondeterminism of the specification is implied by the optionality where a designer selects a better option according to some criteria. We do not rely on machines for having a reset; moreover, if the machines have a reset we still check if it is implemented correctly.

An implementation P *conforms* to the specification S if P is a reduction of S; otherwise, P is a *nonconforming* implementation. According to Proposition 1 the former means that P is isomorphic to some complete submachine of S.

An *adaptive sequence* is an input sequence when the next input of the sequence is selected based on the output of the IUT to the previous inputs. In fact, an adaptive checking sequence is a test case; however, the total length of this test case is big enough and for this reason, we do not talk about the complete test case and usually consider only a part of it that is appropriate for the implementation at hand. Correspondingly, similar to [23], we propose a technique for testing an IUT P on-the-fly but with a single input sequence; the algorithm yields the verdict *pass* if P is a reduction of a given specification FSM S and the verdict *fail* if P is not a reduction of S. Our proposed technique has two procedures. The former checks if P has the same number of states as S and establishes, based on a distinguishing test case, the one-to-one correspondence between the states of P and S if such a correspondence exists. We underline that when such correspondence can be established then an appropriate trace of the distinguishing test case is identified as a state identifier of the corresponding state in P (Proposition 2). The second procedure checks that there is one-to-one correspondence between the transitions of P and an appropriate submachine of S.

4 Deriving an Adaptive Checking Sequence

This section includes two procedures for on-the-fly constructing a checking sequence for a given IUT P from the specification FSM S with respect to the reduction relation. Given a distinguishing case *DTC* for the FSM S, Procedure 1 returns the verdict *fail* if P is not *DTC*-compatible with S; otherwise, it computes the set "state_identifier". For each state *s* of S, this set includes the trace of *DTC* executed by the IUT at the state of P corresponding to *s* and it also includes the state of S reached after this trace. Then Procedure 2 starting from information obtained from Procedure 1 continues deriving the checking sequence where it focuses on checking the one-to-one correspondence between transitions of P and some submachine of S.

As an application example of Procedure 1, consider the implementation FSM P in Fig. 1b and assume that at the beginning of testing P is at state C. After applying *DTC* (in Fig. 2a) we observe a trace *b b*/0 0 which is a trace of state 3 of S. Accordingly, $s = s_1 = 3$, $\sigma = \gamma$. After applying *DTC* again (at Step 1), we observe η = *b b*/0 0 which is a trace of state C of P. Thus, σ becomes *b b b b*/0 0 0 0, $s_2 = 3$, $s' = 3$ as the starting state of S where η is a trace of *DTC*, and the tuple <3, *b b*/0 0, 3> is added to the (initially empty) set "state_identifier", $s = 3$, $\gamma = \eta$ = *b b*/0 0, and then we go to Step 2 as the tuple <3, *b b*/0 0, 3> is in "state_identifier".

Let $s_{new} = 4$, then $s = 4$, we then apply the transfer sequence *c*, observe 1, and thus have η = *c*/1. Then after applying the *b b* of DTC to P, we observe *b b*/0 1. As σ = *b b b b c b b*/0 0 0 0 1 0 1 is a trace at s_1, then we go back to Step 1 where we apply *b a* of DTC and observe 1 0, then s_2 becomes 1, $s' = 2$, and we add the tuple <4, *b b*/0 1, 2> to "state_identifier". Similarly, afterwards, $s = s' = 2$, $\gamma := \eta$ = *b a*, at Step 1, we apply *b a b* of DTC, the trace *b a b*/1 1 1 is observed and the tuple <2, *b a b*/1 0 1, 1> is added to the set "state_identifier".

Then, at Step 2, $s_{new} = s = 1$, after applying $T_{2,1}$ (input sequence *a*) we observe 0 and then after applying again the input sequences *b a* followed by *b b* of DTC the traces *b a*/1 1 followed by *b b*/1 1 are observed and accordingly the tuple <1, *b a* /1 1, 2> is added to "state_identifier". We stop as the set "state_identifier" is complete and the specification FSM reaches state 2 after the observed trace. Table 1 represents the set "state_identifier".

Table 1. The set "state_identifier" for the FSM in Fig. 1b according to the distinguishing test case in Fig. 2.

Current state of S (corresponding state in IUT)	State identifier	Output response	Next state of S (corresponding state in IUT)
1 (A)	*b a*	1 1	2 (B)
2 (B)	*b a b*	1 0 1	1 (A)
3 (C)	*b b*	0 0	3 (C)
4 (D)	*b b*	0 1	2 (B)

Procedure 1: Checking the *DTC*-compatibility between a deterministic IUT P and the specification S

Input: The complete and observable specification FSM S with n states that has a distinguishing test case DTC and a *def*-transfer test case $T_{s,s'}$ for each pair of different states s and s'; a complete deterministic FSM IUT P with at most n states.

Output: Verdict *fail* if P is not *DTC*-compatible with S or state s_2, trace σ, the set "state_identifier".

```
The set "state_identifier" is the empty set;
$\sigma$ is the empty trace;
Apply $DTC$ to P and observe a complete trace $\gamma$ ;
    If $\gamma$ is not a trace at any state of S then
        Return the verdict fail ;
    Else
        Determine state $s_1$ of S with the trace $\gamma$ ;
        $\sigma := \gamma$ ;
        $s := s_1$ ;
Step 1.
Apply $DTC$ to P and observe a complete trace $\eta$ ;
$\sigma := \sigma \eta$ ;
    If $\sigma$ is not a trace at state $s_1$ of S then
        Return the verdict fail ;
    Else
        Determine state $s_2$ where $\sigma$ takes the FSM S from state $s_1$ ;
        Determine state $s'$ of S with the trace $\eta$;
        Add the tuple $< s, \gamma, s' >$ to the set "state_identifier" ;
    If the set "state_identifier" has $n$ tuples then
        End Procedure 1
    Else
        $s := s'$ ;
        $\gamma := \eta$ ;
    If the set "state_identifier" has a tuple $< s, \gamma, s_2 >$ then
        Go-to Step 2 ;
    Else
        Go-back to Step 1 ;
Step 2. Let $s_{new}$ be the state of S such that "state_identifier" has no tuple with the first
item $s_{new}$ ;
$s := s_{new}$;
Apply $T_{s2,s}$ and observe a complete trace $\eta$ ;
Apply $DTC$ to P and observe a complete trace $\gamma$ ;
$\sigma := \sigma \eta \gamma$ ;
    If $\sigma$ is not a trace at state $s_1$ of S then
        Return the verdict fail;
    Else
    Go-back to Step 1;
```

According to Proposition 1, if P is a reduction of S then each state p of P has to have a corresponding state s in S such that the set of complete traces of *DTC* executed at state s has a trace executed at state p, i.e., the following proposition holds.

Proposition 3. If a trace σ observed when executing Procedure 1 is a trace of the specification FSM S, i.e., Procedure 1 does not return the verdict *fail*, then the IUT P is *DTC*-compatible with S. □

If the verdict *fail* is produced by Procedure 1 then the IUT P is not a reduction of S. Otherwise, P is *DTC*-compatible with S and for each state s of S the set "state_identifier" includes the trace of DTC executed by the IUT at the state corresponding to s and it also includes the state of S reached after this trace.

Moreover, if P is *DTC*-compatible with S, then due to Proposition 2, the input projection of a trace observed at state p of P is a state identifier of this state.

```
Procedure 2: Adaptive testing of a deterministic implementation FSM
Input: The complete and observable specification FSM S with $n$ states that has a
       distinguishing test case DTC and a def-transfer test case $T_{s,s'}$ for each pair of
       different states; a complete deterministic FSM P that has at most $n$ states,
       initial state $s_1$, already observed trace $\sigma$ and the set "state_identifier" returned
       by Procedure 1 and state $s_2$ reached after executing Procedure 1.
Output: Verdict pass if and only if P a reduction of S.
The set "Transitions" is the empty set;
$s = s_2$ ;
Step 1.
If there exists input $i$ such that there is no 4-tuple $(s_2, i, o, s'_2)$ with the pair $(s_2, i)$ in
the set "Transitions" then
    Apply $i$ followed by DTC and observe a trace $\eta$ ;
    If $\eta$ is not a trace at state $s_2$ then
        Return the verdict fail ;
    Else
        Add the transition $(s_2, i, o, s')$ to the set "Transitions" where $o$ is the ob-
        served output to $i$ while $s'$ is the current state according to the set
        "state_identifier";
        $\sigma := \sigma\eta$ ;
        $s_2 := s'$ ;
        Go-back to Step 1;
Else
Step 2.
If the set "Transitions" is full, i.e., the set includes an item $(s, i, o, s')$ for every pair
$(s, i)$ then
    Return the verdict pass ;
Else
    Using transitions of the set "Transitions" determine a trace $\eta$ from $s_2$ to state $s$
    such that for some $i$ there is no transition $(s, i, o, s')$ with the pair $(s, i)$ in the
    set "Transitions" ;
    $\sigma := \sigma\eta$ ;
    $s_2 := s$ ;
Go-back to Step 1;
```

Proposition 4. Let the IUT P be *DTC*-compatible with S, i.e., there exists one-to-one correspondence F: $S \rightarrow P$ such that for each state $s \in S$ it holds that the intersection of $Tr(\mathrm{S}/s) \cap Tr(\mathrm{P}/p) \cap CompleteTraces(\mathrm{TC})$ is not empty if and only if $p = F(s)$. Given a tuple <s, γ, s'> of the set "state_identifier", the input projection of trace γ is a state identifier of state $F(s)$ while s' is the state of S reached by γ. □

Proposition 5. The verdict *pass* is produced by Procedure 2 if and only if the IUT P is a reduction of S.

Proof. If FSM P passes Procedure 1 then P is strongly connected, since each state of P is traversed when executing Procedure 1. For this reason, if at Step 1, the reached state has no unchecked transitions then at Step 2, in the set "Transitions" that has only already checked transitions, there is a path to a state with an unchecked transition. Procedure 2 establishes the one-to-one correspondence between transitions of FSM P and an appropriate submachine of S, since all P transitions are executed and checked for a conforming output and corresponding final state (according to DTC). Therefore, FSM P passes Procedure 2 if and only if P is isomorphic to some submachine of S and the proposition holds according to Proposition 1. □

As an application example, consider the FSM in Fig. 1a, after applying Procedure 1, we obtain $s_1 = 3$, the trace σ, and state $s_2 = 2$ (reached after applying σ); in addition, the set "state_identifier" = {<1, *b a* /1 1, 2>, <2, *b a b*/1 0, 1>, <3, *b b*/0 0, 3>, <4, *b b*/ 0 1, 2 >}. As $s = s_2 = 2$ and the set "Transitions" is empty, at Step 1, apply the input *b* at s_2 followed by *b a b* of *DTC* and observe the trace *b b a b*/1 1 0 1 that reaches state 1 of the FSM S. Add (2, *b*, 1, 2) to "Transitions", σ becomes that of Procedure 1 concatenated with the trace *b b a b*/1 1 0 1, and the reached state $s_2 = 1$ according to the tuple <2, *b a b*/1 0 1, 1> of the "state_identifier". We go-back to Step 1, apply the input *a* followed by *b a b* of DTC and observe 1 1 0 1, reach state $s_2 = 1$, add (1, *a*, 1, 2) to "Transitions", append σ as usual, and proceed again to Step 1. At $s_2 = 1$ apply *b*, then apply *b a* of *DTC* and observe 1 1 1; add (1, *b*, 1, 1) to "Transitions" and reach state $s_2 = 2$. Again at Step 1, apply *a* followed by *b a* and observe 0 1 1, add (2, *a*, 0, 1) to "Transitions". Then at the reached state $s_2 = 2$, apply *c* followed by *b b* of *DTC*, observe 1 0 1, add (2, *c*, 1, 4) to "Transitions" and reach state $s_2 = 2$. Now, as all (2, *b*, 1, 2), (2, *a*, 0, 1), (2, *c*, 1, 4) are in the set "Transitions", at Step 2, we consider $s = 4$ such that from the reached state $s_2 = 2$ there is the checked trace $\eta = c/1$ from 2 to 4 and for some input *i* there is no transition (4, *i*, *o*, *s*′) in the set "Transitions". We transfer to state 4 from state 2 by applying the input *c*, now s_2 becomes the reached state 4. We go-back to Step 1 where we select to apply the input *a* followed by applying the sequence *b b* of *DTC*, observe 1 0 0, add (4, *a*, 1, 3) to "Transitions" and reach state $s_2 = 3$. We proceed as above till the set "Transitions" is full. The verdict *pass* is produced after completing the set "Transitions" and thus, FSM P is a reduction of the specification FSM S.

As the length of a transfer sequence when checking a new input is less than the number n of states of the FSM S, the length of a checking sequence returned by Procedure 2 is proportional to the length of a distinguishing test case. If this length is polynomial with respect to the number of S states as it happens for merging-free FSMs then the length of an adaptive checking sequence is O(n^3), i.e., the length evaluation is almost similar to that for deterministic FSMs [10].

5 Conclusion

In this paper, we have proposed an adaptive strategy for testing a deterministic implementation FSM against nondeterministic observable specification FSM with respect to the reduction relation. Similar to deterministic FSMs, the strategy can be applied under appropriate restrictions upon the specification FSM and fault domain. However, we show that the requirement of the existence of a separating sequence can be replaced by the requirement of the existence of a distinguishing test case. This is useful as the existence of a distinguishing test case is more likely than that of a separating sequence and generally, the length of a distinguishing test case is less than that of a separating sequence (when both exist). In addition, the construction uses adaptive transfer sequences that reduce the length of an applied input sequence. We note that in this paper, we do not discuss any optimization procedure for deriving adaptive checking sequences; this is left for the future work. Another possible direction of a future work is the extension of the proposed work for testing nondeterministic non-initialized implementations. It could be also interesting to apply a proposed approach for deriving checking sequences for I/O automata, for example, with respect to the widely used *ioco* conformance relation that is very close to the reduction relation between FSMs.

Acknowledgment. This work was partially supported by the RSF project №. 16-49-03012.

References

1. Alur, R., Courcoubetis, C., Yannakakis, M.: Distinguishing tests for nondeterministic and probabilistic machines. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, pp. 363–372 (1995)
2. Bochmann, G.V., Petrenko, A.: Protocol testing: review of methods and relevance for software testing. In: Proceedings of the ACM International Symposium on Software Testing and Analysis, pp. 109–123 (1994)
3. Chow, T.S.: Testing software design modelled by finite state machines. IEEE Trans. Softw. Eng. **4**(3), 178–187 (1978)
4. Dorofeeva, R., El-Fakih, K., Maag, S., Cavalli, A., Yevtushenko, N.: FSM-based conformance testing methods: a survey annotated with experimental evaluation. Inf. Softw. Technol. **52**(12), 1286–1297 (2010)
5. El-Fakih, K., Yevtushenko, N., Kushik, N.: On the reachability of the exponential upper bound of adaptive experiments for nondeterministic finite state machines (2016, submitted)
6. Ermakov, A.: Deriving checking sequences for nondeterministic FSMs. In: Proceedings of the Institute for System Programming of RAS, vol. 26, pp. 111–124 (2014). (In Russian)
7. Gonenc, G.: A method for the design of fault detection experiments. IEEE Trans. Comput. **19**(6), 551–558 (1970)

8. Güniçen, C., Jourdan, G.-V., Yenigün, H.: Using multiple adaptive distinguishing sequences for checking sequence generation. In: El-Fakih, K., et al. (eds.) ICTSS 2015. LNCS, vol. 9447, pp. 19–34. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25945-1_2
9. Güniçen, C., Inan, K., Türker, U.C., Yenigün, H.: The relation between preset distinguishing sequences and synchronizing sequences. Formal Aspects Comput. **26**(6), 1153–1167 (2014)
10. Hennie, F.C.: Fault-detecting experiments for sequential circuits. In: Proceedings of the Fifth Annual Symposium Switching Circuit Theory and Logical Design, pp. 95–110 (1964)
11. Hierons, R.M., Ural, H.: Reduced length checking sequences. IEEE Trans. Comput. **51**(9), 1111–1117 (2002)
12. Hierons, R.M., Ural, H.: Optimizing the length of checking sequences. IEEE Trans. Comput. **55**(5), 618–629 (2006)
13. Kohavi, Z.: Switching and Finite Automata Theory. McGraw-Hill, New York (1978)
14. Petrenko, A., Simão, A., Yevtushenko, N.: Generating checking sequences for nondeterministic finite state machines. In: Proceedings of the International Conference on Software Testing, pp. 310–319 (2012)
15. Kushik, N: Methods for deriving homing and distinguishing experiments for nondeterministic FSMs. Ph.D. thesis, Tomsk State University (2013). (In Russian)
16. Yevtushenko, N., Kushik, N.: Decreasing the length of adaptive distinguishing experiments for nondeterministic merging-free finite state machines. In: Proceedings of IEEE East-West Design & Test Symposium, pp. 338–341 (2015)
17. Kushik, N., El-Fakih, K., Yevtushenko, N.: Adaptive homing and distinguishing experiments for nondeterministic finite state machines. In: Yenigün, H., Yilmaz, C., Ulrich, A. (eds.) ICTSS 2013. LNCS, vol. 8254, pp. 33–48. Springer, Heidelberg (2013)
18. Yevtushenko, N., Kushik, N., El-Fakih, K., Cavalli, A.R.: On adaptive experiments for nondeterministic finite state machines. Int. J. Softw. Tools Technol. Transf. **18**(3), 251–264 (2016)
19. Kushik, N., Yevtushenko, N., Yenigun, H.: Reducing the complexity of checking the existence and derivation of adaptive synchronizing experiments for nondeterministic FSMs. In: Proceedings of the International Workshop on Domain Specific Model-Based Approaches to Verification and Validation (AMARETTO 2016), pp. 83–90 (2016)
20. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines-a survey. Proc. IEEE **84**(8), 1090–1123 (1996)
21. Petrenko, A., Simão, A.: Generalizing the DS-methods for testing non-deterministic FSMs. Comput. J. **58**(7), 1656–1672 (2015)
22. Petrenko, A., Yevtushenko, N.: Conformance tests as checking experiments for partial nondeterministic FSM. In: Grieskamp, W., Weise, C. (eds.) FATES 2005. LNCS, vol. 3997, pp. 118–133. Springer, Heidelberg (2006)
23. Petrenko, A., Yevtushenko, N.: Adaptive testing of deterministic implementations specified by nondeterministic FSMs. In: Wolff, B., Zaïdi, F. (eds.) ICTSS 2011. LNCS, vol. 7019, pp. 162–178. Springer, Heidelberg (2011)
24. Simao, A., Petrenko, A., Maldonado, J.C.: Comparing finite state machine test. IET Softw. **3** (2), 91–105 (2009)
25. Spitsyna, N., El-Fakih, K., Yevtushenko, N.: Studying the separability relation between finite state machines. Softw. Test. Verification Reliab. **17**(4), 227–241 (2007)
26. Lee, D., Yannakakis, M.: Testing finite-state machines: state identification and verification. IEEE Trans. Comput. **43**(3), 306–320 (1994)