Adaptive Testing of Nondeterministic Systems with FSM

Alexandre Petrenko
CRIM, Centre de recherche informatique de Montréal, 405 Ogilvy Avenue, Suite 101 Montréal (Québec) H3N 1M3, Canada
petrenko@crim.ca

Nina Yevtushenko
Tomsk State University, 36 Lenin Street, Tomsk, 634050, Russia
ninayevtushenko@yahoo.com

Abstract—The paper addresses the problem of testing systems considered as black boxes and modelled by Finite State Machines (FSM). While the theory of testing from deterministic FSMs has received a lot of attention, testing from nondeterministic machines is not yet investigated to the same extend. The paper is devoted to the problem of generation of adaptive test cases for nondeterministic specification FSMs to obtain complete, i.e., sound and exhaustive, test suites. Such tests verify that an implementation conforms to its specification, when the conformance relation is the reduction (trace inclusion) relation. A method for deriving a complete test suite is proposed. In case, when the implementation under test is also nondeterministic, each test case is adaptively executed a predefined number of times ensuring that all nondeterministic choices were taken. Comparison with the existing approaches indicates that the method has several advantages.

Keywords—formal methods; test generation; adaptive testing; conformance testing; nondeterministic FSM

I. INTRODUCTION

Formal methods are indispensable in high assurance systems engineering, as they provide repeatable results in verification and testing: one can always take models and reproduce a detected flaw to make sure it is not a false alarm or that it is fixed. Model-based testing is a very active research area, where several academic and commercial tools are already being applied for safely-critical systems. It is often argued that well-engineered systems are deterministic, so testing should focus on such systems. However, nondeterministic models are often more adequate to represent the behavior of complex systems. Nondeterminism may occur for various reasons.

The problem of model-based testing in the context of nondeterministic reactive systems has traditionally been addressed with state-oriented models, such as finite state machines and transition systems, both distinguishing inputs and outputs. Inputs to an Implementation Under Test (IUT) are generated by a tester, while outputs are analyzed by the latter to conclude whether the IUT is a conforming implementation, and to be exhaustive it must fail on each nonconforming implementation. Obtaining and executing such a test suite is a challenging problem in the presence of nondeterminism. The issue is that in testing an IUT is treated as a black box and if it is not deterministic then executing a test case the tester has to be assured that the IUT has shown all the reactions to the test case, so the tester has a complete test observation. Correspondingly, there is a universally accepted assumption that an IUT behaves “fairly” during testing. This is a so-called “all-weather conditions” [8] or complete testing assumption [7]. In practical terms, it means that for an IUT from a particular application domain, the tester knows under which conditions test cases must be re-executed to have a sufficient confidence about the completeness of the observed reactions of the IUT.

Research in conformance testing from Input/Output FSMs (Mealy machines) focuses on two conformance relations, trace equivalence and trace inclusion, called the reduction relation between nondeterministic FSMs [10]. Most approaches for test derivation are fault-model based, allowing to claim the completeness of a resulting finite test suite with respect to a fault model, chosen for a given conformance relation. These approaches generalize the classical work on checking experiments for deterministic FSMs [9] [16] to the class of nondeterministic machines.

We focus in this paper on the problem of adaptive test case generation from nondeterministic FSMs to obtain complete test suites with respect to the reduction (trace inclusion) relation, allowing the IUT to behave nondeterministically. The completeness of a test suite is proven for a fault model, which includes all possible implementations within a predefined bound on the number of states, which may even exceed that of the specification FSM.

The remaining of this paper is organized as follows. Section 2 presents basic definitions and notions for state machines. Section 3 explains the type of test cases used by the proposed approach and defines an m-complete test suite. Section 4 details the approach for deriving such a test suite for adaptive execution. The related work is discussed in Section 5 which concludes the paper.
II. GENERAL DEFINITIONS

A Finite State Machine (FSM) $S$ is a $5$-tuple $(S, s_0, I, O, h)$, 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; $h$ is a transition relation $h \subseteq S \times I \times O \times S$, where a $4$-tuple $(s, i, o, s') \in h$ is a transition.

Input sequence $\alpha \in I^*$ is a defined input sequence in state $s$ of $S$ if it labels a sequence of transitions starting in state $s$. A trace of $S$ in state $s$ is a string of input-output pairs which label a sequence of transitions starting in state $s$. Let $\text{Tr}(s)$ denote the set of all traces of $S$ in state $s$, while $\text{Tr}$ denote the set of traces of $S$ in the initial state. Given sequence $\beta \in (I\cup O)^*$, the input (output) projection of $\beta$, denoted $\mathbf{\text{projection}}(\beta)$, is a sequence obtained from $\beta$ by erasing symbols in $O(I)$.

We define various types of machines as follows.

FSM $S = (S, s_0, I, O, h)$ is

- trivial if $h = \emptyset$;
- completely specified (complete FSM) if for each pair $(s, i) \in S \times I$ there exists $(o, s') \in O \times S$ such that $(s, i, o, s') \in h$;
- partially specified (partial FSM) if for some pair $(s, i) \in S \times I$, input $i$ is undefined in state $s$, i.e., $(s, i, o, s') \notin h$ for all $(o, s') \in O \times S$;
- deterministic (DFSM) if for each pair $(s, i) \in S \times I$ there exist at most one transition $(s, i, o, s') \in h$ 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 $(s, i, o_1, s_1), (s, i, o_2, s_2) \in h$, such that $o_1 \neq o_2$ or $s_1 \neq s_2$;
- observable if for each two transitions $(s, i, o, s_1), (s, i, o, s_2) \in h$ 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 $(s, i_1, o_1, s_1), (s, i_2, o_2, s_2) \in h$ 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 $\text{Tr}$ is finite.

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

In this paper, we consider only completely connected observable specification machines; one could use a standard procedure for automata determinization to convert a given FSM into observable one.

To characterize the common behavior of two machines (states) we use the operation of the intersection. The intersection $S \cap P$ (also known as the product) of two machines $S = (S, s_0, I, O, h)$ and $P = (P, p_0, I, O, h)$ is an FSM $(Q, q_0, I, O, h_{\cap})$ with the state set $Q \subseteq S \times P$, the initial state $q_0 = s_0p_0$, and the transition relation $h_{\cap}$ such that $Q$ is the smallest state set obtained by using the rule $(i, i', o, s', p') \in h_{\cap} \iff (s, i, o, s') \in h_1$ and $(p, i, o, p') \in h_2$. The intersection $S \cap P$ preserves only common traces of the two machines, in other words, $\text{Tr}_{\cap} = \text{Tr}_1 \cap \text{Tr}_2$. We will also use the fact that one machine is a submachine of another in the following sense. FSM $P$ is a submachine of $S$ if $P \subseteq S, p_0 = s_0$ and $h \subseteq h_1$.

We define in terms of traces several relations between states of a complete FSM.

Given states $s_1, s_2$ of a complete FSM $S = (S, s_0, I, O, h)$, let $S_1 = (S, s_1, I, O, h)$ and $S_2 = (S, s_2, I, O, h)$, then

- $s_1$ and $s_2$ are (trace-) equivalent, if $\text{Tr}_1 = \text{Tr}_2$;
- $s_1$ and $s_2$ are trace-included into (is a reduction of) $s_1, s_2 \subseteq s_1$, if $\text{Tr}_2 \subseteq \text{Tr}_1$;
- $s_1$ and $s_2$ are r-compatible, $s_1 \simeq s_2$, if the intersection $S_1 \cap S_2$ has a complete submachine;
- $s_1$ and $s_2$ are r-distinguishable, $s_1 \not\simeq s_2$, if the intersection $S_1 \cap S_2$ has no complete submachine.

We also use relations between machines. Given FSMs $S = (S, s_0, I, O, h)$ and $P = (P, p_0, I, O, h)$, $P \preceq S$ if $s_0 \preceq p_0, P \simeq S$ if $s_0 = p_0, P \not\simeq S$ if $s_0 \not\simeq p_0, P \not\simeq S$ if $s_0 \not\simeq p_0$.

III. TEST CASES

When deriving tests against FSMs usually a test suite is a finite set of finite input sequences of the specification FSM [2] and each input sequence is a test case. When talking about adaptive test execution we should consider adaptive test cases. Depending on the observed output reaction to the previous input, the tester either chooses a next input to execute depending on a produced output or terminates if there are no more inputs to execute in the test case. Any unexpected output triggers in the test case a transition to the state fail, terminating test execution. In order to allow a tester to have a complete control over the test execution only one input should be defined at each state of a test case. Correspondingly, we come to the following definition of a test case.

Definition 1. A test case over input alphabet $I$ and output alphabet $O$ is an acyclic single-input output-complete FSM $\mathcal{U}$
A test case is **trivial** if it is a trivial FSM with the initial state that is not the **fail** state. By definition, a test case not necessary has a fail state; in this case, every implementation FSM passes this test case. A trivial FSM is an example of such a test case.

Given a test case $U$, we further refer to traces which take $U$ from the initial state to the fail state as **fail** traces. **Pass** traces are defined as completed traces of $U$ that do not lead $U$ to the fail state.

A test case is **preset** if all input projections of its pass traces are prefixes of a single input sequence, otherwise it is **adaptive**.

Given a complete FSM $S = (S, s_0, I, O, h)$, let $\mathcal{I}(S)$ be a set of complete possibly nondeterministic (implementation) machines over the input alphabet $I$ and the output alphabet $O$, called a **fault domain**. FSM $\mathcal{B} \in \mathcal{I}(S)$ is a **conforming** implementation machine of $S$ w.r.t. the reduction relation if $\mathcal{B} \leq S$.

**Definition 2.** Given the specification FSM $S$, a test case $U = (U, u_0, I, O, h)$, and an implementation FSM $\mathcal{B} \in \mathcal{I}(S)$,

- $\mathcal{B}$ passes the test case $U$, if the intersection $\mathcal{B} \cap U$ has no state, where the test $U$ is in the state fail.
- $\mathcal{B}$ fails $U$ if the intersection $\mathcal{B} \cap U$ has a state, where the test $U$ is in the state fail.

A test suite $TS$ is

- sound for FSM $S$ in $\mathcal{I}(S)$ w.r.t. the reduction relation, if any $\mathcal{B} \in \mathcal{I}(S)$, which is a reduction of $S$ passes each test case of the TS.
- exhaustive for FSM $S$ in $\mathcal{I}(S)$ w.r.t. the reduction relation, if for any $\mathcal{B} \in \mathcal{I}(S)$, which is not a reduction of $S$ there exists a test case $U$ in $TS$ such that $\mathcal{B}$ fails the $U$.
- complete for FSM $S$ in $\mathcal{I}(S)$ w.r.t. the reduction relation, if it is sound and exhaustive for FSM $S$ in $\mathcal{I}(S)$ w.r.t. the reduction relation.

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

Similar to the concatenation operator over sequences we define the concatenation operator over test cases. Given test cases $TC_1$ and $TC_2$ and a dead洛克 state $t \neq \text{fail}$ of $TC_1$, we define the concatenation $TC_1 \circ TC_2$ as a test case where state $t$ of $TC_1$ is identified with the initial state of $TC_2$. Correspondingly, each completed trace that takes $TC_1$ from the initial state to the state $t$ is continued with each trace of the test case $TC_2$. The operator $\circ$, is associative, i.e., we simply write, e.g., $TC_1 \circ TC_2 \circ TC_3$ when concatenating three test cases.

Notice that in order to conclude that the implementation passes the test case for the reduction relation, we rely on the complete testing assumption and assume that given any FSM $\mathcal{B} \in \mathcal{I}_m(S)$ that fails a test case $TC$, the $TC$ is applied to $\mathcal{B}$ an appropriate number of times in order to observe a fail trace.

IV. **TEST GENERATION METHOD**

In this section, we develop a method for building $m$-complete test suites by using the concatenation of the following test fragments: state preambles needed to reach states, state identifiers for checking the reached state and trace traversal sets, which allow to execute transitions and to reach additional states if they are present in a given implementation. These fragments can be seen as generalization of the three types of input sequences in all the methods for complete test suite generation from deterministic machines [2][3][14]. Those are transfer, state identification and traversal sequences.

A. **State Preambles**

The following definitions are taken from [14].

**Definition 3.** Given an FSM $S = (S, s_0, I, O, h)$, state $s \in S$ is definitely reachable if any complete reduction of $S$ has a trace which takes $S$ into the state $s$.

This property can be established as follows.

**Proposition 1.** State $s$ of an FSM $S$ is definitely reachable if and only if $S$ has a single-input acyclic submachine $S'$ with the only deadlock state $s$ such that for each input defined in some state of $S'$, the state has all the transitions of $S$ 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 4.** Given a definitely reachable state $s \in S$, a single-input acyclic submachine of $S$ 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 $S$ labeled with this input is a preamble for state $s$, denoted $P_s = (R, r_0, I, O, h_s)$.

Given a preamble $P_s$, we derive a test case $TC(P_s)$ by augmenting the preamble $P_s$ to an output-complete FSM, namely, we add all transitions from every state with missing outputs to a designated deadlock state $\text{fail}$.

B. **State Separators**

The notion of a separator for two states of a given NFSM can be considered as the generalization of the notion of separating sequence used for the equivalence relation of DFSM, to the reduction relation.

**Definition 5.** Given $r$-distinguished states $s_1$ and $s_2$ of a complete FSM $S$, a single-input acyclic FSM $R(s_1, s_2) = (R, r_0, I, O, h)$ with the deadlock states $\bot$ and $\bot$ is a separator $R(s_1, s_2)$ of states $s_1$ and $s_2$ if the following two conditions hold:
\[ \bullet r_{α}\text{-after-}α = \bot \text{ implies } α \in Tr(s_i) / Tr(s_j) \text{ and } r_{β}\text{-after-}α = \bot \text{ implies } α \in Tr(s_i) / Tr(s_j); \]

\[ \bullet \text{ for each trace } α \text{ of } R \text{ and input } x \text{ defined in } r_{α}\text{-after-}α, \]
\[ \text{ out}(r_{α}\text{-after-}α, x) = \text{ out}(S_{α}\text{-after-}α, x) \cup \text{ out}(S_{β}\text{-after-}α, x). \]

For the method for constructing separators we refer the reader to [1] and [14]. A separator \( R(s_i, s_j) \) allows us to identify a current state of the FSM \( S \) among the two, \( s_i \) or \( s_j \). To check whether an implementation is in a state that behaves as only one of them, we use a separator of state \( s_i \), from state \( s_j \).

Given a separator \( R(s_i, s_j) \) of \( r\)-distinguishable states \( s_i \) and \( s_j \), we derive a test case \( R(s_i) \) distinguishing \( s_j \) from state \( s_i \) by replacing the state \( \bot \) with the designated state \( \text{fail} \) and augmenting the obtained FSM to an output-complete FSM by adding all transitions from every state with missing outputs to the fail state.

Definition 6. Given state \( s_i \) of \( S \) let \( R(s_i) \) be the set of all states \( r\)-distinguishable from \( s_i \). The set \( \{ R(s_i) \mid s_j \in R(s_j) \} \) is an identifier of state \( s_i \), denoted \( ID_{r,s_i} \).

C. Traversal Sets

The construction of traversal sets is based on the approach elaborated in [12] and we refer the reader to that work for more details. The basic idea is to count states of the specification FSM traversed by a trace and to terminate the trace as soon as it becomes cyclic in any conforming implementation FSM with at most \( m \) states. The termination rule is formulated using the number of pairwise \( r\)-distinguishable states traversed by a trace started at some definitely reachable state.

For each set \( R \) of pairwise \( r\)-distinguishable states of the specification FSM \( S \), the set \( R_{α} \) denotes a subset of definitely reachable states of \( R \), while \( R \) is the set of all maximal sets of pairwise \( r\)-distinguishable states. If a state \( s \) is \( r\)-compatible with any state of \( S \) then the set \( R \) has a singleton \( \{ s \} \). Moreover, if \( s \) is definitely reachable then \( R_{α} = \{ s \} \), otherwise \( R_{α} = \emptyset \). In fact, the set \( R_{α} \) can be empty even if \( |R| > 1 \).

Given a definitely reachable state \( s \), two sets are simultaneously constructed, the sets \( N^m(s) \) and \( T^m(s) \). \( N^m(s) \) contains input sequences, called \( m\)-traversal sequences and is constructed starting from the empty sequence by adding inputs one by one with the following termination rule. A current input sequence \( α \) is not extended any longer as soon as for each trace \( β \in Tr(s) \), \( β | α = α \), there exists \( R \in R_{α} \) such that the trace \( β \) started at state \( s \) traverses states of the set \( R \) \((m - |R_{α}| + 1) \) times. We say that trace \( β \) traverses state \( s′ \) if \( S\text{-after-}β′ = s′ \) for some non-empty prefix \( β′ \) of \( β \).

The set \( T^m(s) \) contains traces of the FSM \( S \) and is constructed starting from the empty sequence by adding traces as follows. Given an input sequence \( α \in N^m(s) \), a trace \( β \) is included into the set \( T^m(s) \) if \( β | α = α \) and for each \( R \in R_{α} \), it traverses states of the set \( R \) at most \((m - |R_{α}|) \) times or there exists \( R \in R_{α} \) such that \( s\text{-after-}β \in R \) and trace \( β \) traverses states of the set \( R \) in the FSM \( S \) exactly \((m - |R_{α}| + 1) \) times. Recall that in case of a minimal initially connected DFSM with \( n \) states, each state is definitely reachable and hence \( R_{α} = n \).

The set \( T^m(s) \) is the trace traversal set.

Given \( α \in N^m(s) \), let \( \{ β \in T^m(s) \mid β | α \in \text{pref}(α) \} = \text{Traces}(s, α) \). We first build an acyclic observable FSM with the set of traces \( \text{Traces}(s, α) \), such that any two completed traces converge (lead to the same state) only if they converge in the FSM \( S \). Since we consider all the traversal traces with the input projections which are prefixes of \( α \in N^m(s) \), to simplify the machine, we delete such traversal traces. Namely, if there exists a trace \( β \), such that for every completed trace \( β γ \) the input projection is a proper prefix of \( α \) then we delete all such completed traces and add the completed trace \( β \).

Then we convert the obtained FSM into a test case \( TC(s, α) \) by augmenting it, if needed, to an output-complete FSM, i.e., we add all transitions from every state with missing outputs to a \( \text{fail} \) state.

D. Building a complete test suite

In this section, we present an algorithm for deriving an \( m\)-complete test suite using the concatenation of the defined test fragments. Similar to a preset test suite construction, the state reached in the IUT after a preamble \( P_p \) is checked using the corresponding state identifier. This is achieved by building a separate test case for each state separator of the state identifier. If the observed traces of the obtained test cases are as expected then the implementation under test reaches a state that cannot be a reduction of any state of the specification that is \( r\)-distinguishable with state \( s \). Each preamble \( P_p \) is also concatenated with each test case \( TC(s, α) \) where \( α \) is an \( m\)-traversal sequence. Each state of the IUT reached after an observed trace with the input projection \( α \) is again checked as above. All the traces with the prefixes of the input sequence \( α \) are in the traversal set \( N^m(s) \) and the reached states are checked in the corresponding test cases.

Algorithm 1 for deriving an \( m\)-complete test suite

Input. Complete FSM \( S \), for each definitely reachable state \( s \), a test case \( TC(P_p) \) for the preamble \( P_p \) and the set of test cases \( TC(s, α) \), where \( α \in N^m(s) \); for each \( s \in S \), an identifier \( ID_{s} \).

Output. An \( m\)-complete test suite.

Initialize the empty set \( TS \);

For each definitely reachable state \( s \) and each test case \( TC(s, α) \),

Let \( \{ β_1, \ldots, β_n \} \) be the set of all pass traces of length \(|α| \) of \( TC(s, α) = (T, t_0, I, O, τ_{χ}, τ_{β}, τ_{γ}) \);

Determine a set \( N_α \) of test cases \( TC(s, α)@q_{τ_{β}⇒q_{τ_{β}}} \) where \( R_{τ_{β}⇒q_{τ_{β}}} \in ID_{s\text{-after-}β} \) such that each test case from any identifier is concatenated once for each trace \( β_i \);

\[ TS := TS \cup \{ TC(P_p)@TC \mid TC \in N_α \}; \]
The main contribution of this paper is an approach for generating an m-complete test suite from NFSM, as a set of tests, where a test is a single-input output-complete acyclic FSM. The execution of the entire test suite is completely adaptive, reducing the cost of testing. The proposed approach exploits the fact that the specification could have not only deterministically, but also definitely reachable states. This allows reducing the test length compared to the existing methods. In our previous work on test generation from NFSM, we used a compact representation of an m-complete test as a single machine and did not require a test be a single-input machine. We notice that in all known methods for test generation from FSM, including our previous work, which use transfer and traversal sequences, these input sequences are executed in a non-adaptive way. In the proposed approach their execution is fully adaptive.

REFERENCES