CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Brunel University Research Archive

# Adaptive testing of a deterministic implementation against a nondeterministic finite state machine

R.M. Hierons

1998

**abstract** A number of authors have looked at the problem of deriving a checking experiment from a nondeterministic finite state machine that models the required behaviour of a system. We show that these methods can be extended if it is known that the implementation is equivalent to some (unknown) deterministic finite state machine.

When testing a deterministic implementation, the test output provides information about the implementation under test and can thus guide future testing. The use of an adaptive test process is thus proposed.

## 1 Introduction

*Nondeterministic finite state machines* (*NFSMs*) are used to model a number of types of computer system including communications protocols (Tanenbaum [1996]). If an NFSM model $M$, of the required behaviour, exists and some implementation $I$ has been produced, it is important to verify $I$ against $M$. Testing usually forms part of the verification process. When testing it is normal to assume that the *implementation under test* (*IUT*) $I$ behaves like some (possibly nondeterministic) finite state machine $M_I$.

As nondeterminism can aid abstraction, many specifications are nondeterministic. Actual implementations are, however, often deterministic. The situation, in which $M$ is nondeterministic and $M_I$ is deterministic, is thus of interest.

An NFSM $M$ is defined by a tuple $(S, s_1, h, X, Y)$ in which $S$ is a finite set of *states*, $s_1$ is the *initial state*, $X$ is the finite *input alphabet*, $Y$ is the finite *output alphabet* and $h$ is the *state transition function*. Given an NFSM $M$, $S_M$ shall denote the state set of $M$. When $M$ receives an input value $x \in X$, while in state $s \in S$, a *transition* is executed producing an output value $y \in Y$ and moving $M$ to some state $s' \in S$. The function $h$ gives the possible transitions and has type $h : S \times X \rightarrow \mathcal{P}(S \times Y)$, where $\mathcal{P}$ denotes the power set operator. Thus, if $M$ receives input $x \in X$ while in state $s \in S$, the output $y \in Y$ and next state $s' \in S$ satisfy $(s', y) \in h(s, x)$.

An NFSM, that shall be denoted $M_0$ throughout this paper, is given in Figure 1. Here, $S = \{s_1, \ldots, s_5\}$, $X = \{a, b\}$ and $Y = \{0, 1\}$. If $b$ is input while $M_0$ is in state $s_4$, either 0 is output and $M_0$ moves to state $s_5$ or 1 is output and $M_0$ moves to state $s_2$. Thus $h(s_4, b) = \{(s_5, 0), (s_2, 1)\}$.

A *finite automaton* (*FA*) $N$ is defined by a tuple $(S, s_1, \gamma, \Sigma, F)$, in which $S$ is a finite set of *states*, $s_1 \in S$ is the initial state, $\gamma$ is the *state transition function*, $\Sigma$ is the finite *alphabet* and $F \subseteq S$ is the set of *final states*. The function $\gamma$ takes a state $s \in S$ and a value $x \in \Sigma$ and gives a set of possible next states $\gamma(s, x)$. Let $\epsilon$ denote the empty sequence and, given a set $\Gamma$, $\Gamma^*$ denote the set of strings consisting of elements of $\Gamma$. Thus $\epsilon \in \Gamma^*$. The transition function $\gamma$ can be extended, to take values from $\Sigma^*$, giving the function $\gamma^*$ defined by: $\gamma^*(s, \epsilon) = \{s\}$, $\gamma^*(s, yx) = \{s' | (\exists s'' \in \gamma^*(s, y) \bullet s' \in \gamma(s'', x))\}$ (where $x \in \Sigma$ and $y \in \Sigma^*$).

FA are used to define languages: a string $x \in \Sigma^*$ is *accepted* by $N$, and thus is part of the language defined by $N$, if and only if $\gamma^*(s_1, x) \cap F \neq \emptyset$. The FA $N$ is deterministic if for each state $s \in S$ and input value $x \in \Sigma$, there is only one possible next state. Given a nondeterministic FA there is some deterministic FA that defines the same language. There are standard algorithms for finding such an equivalent deterministic FA (Rabin and Scott [1959]).

An NFSM $M$ can be thought of as a FA $F(M)$ in which the elements of $\Sigma$ are the input/output pairs and all the states are final states. The NFSM $M$ is then characterized by the language defined by $F(M)$: the set of input/output sequences that $M$ allows. This set shall be denoted $L(M)$.

If $L(M_I) \subseteq L(M)$, $M_I$ is said to be a *reduction* of $M$ and this is denoted $M_I \leq M$. One definition of conformance is that $I$ *conforms* to $M$ if $M_I \leq M$. Testing can thus be characterized as trying to determine whether $I$ conforms to $M$. This definition of conformance matches the notion of correctness used in (Dick and Faivre [1993], Hierons [1997a]) and will be used throughout the



BURA

rest of the paper.

A transition is defined by its initial state, final state, input and output. Given a sequence $t$ (possibly of length 1) of transitions, $t^{in}$ will denote the input sequence from $t$, $t^{out}$ will denote the output sequence produced by $t$, $start(t)$ will denote the initial state of $t$ and $end(t)$ will denote the final state of $t$.

From the function $h$, the next state function $h_1$ and output function $h_2$ can be derived. Thus if $M$ receives input $x$ while in state $s$ the next state is one of those contained in $h_1(s,x)$ and the output is one of the values from $h_2(s,x)$. The functions $h$, $h_1$, and $h_2$ can be extended, in a similar manner to $\gamma$, to take a state and an input sequence giving functions $h^*$, $h_1^*$, and $h_2^*$ respectively.

An NFSM $M$ is *completely specified* if, for each $s \in S$ and $x \in X$, $|h(s,x)| \geq 1$ and $M$ is *deterministic* if for each $s \in S$ and $x \in X$, $|h(s,x)| \leq 1$. If $M$ is deterministic and completely specified the transitions are completely defined by a *next state function* $\delta$ and an *output function* $\lambda$. A *deterministic finite state machine* (DFSM) can thus be defined by the tuple $(S, s_1, \delta, \lambda, X, Y)$.

It should be noted that, unlike FA, there are NFSM for which there is no equivalent DFSM. Consider, for example, the response of $M_0$ to the input of $a$. There are two possible output values: 0 and 1. This behaviour cannot be represented by a DFSM.

For each $y \in Y$, $s \in S$, and $x \in X$, $h_y(s,x)$ will denote the set of possible next states if a transition is executed from $s$ using input $x$ and produces output $y$. Again, this can be extended to $h_y^*$. An NFSM $M$ is said to be *observable* if for every $s \in S$, $x \in X$, and $y \in Y$, $|h_y(s,x)| \leq 1$. Consider, for example, the response of $M_0$ to the input value $a$ while in state $s_1$. Although there are two possible behaviours, they have different output values. Thus, when the output is known, there is only one possible next state. This reduces the uncertainty caused by nondeterminism.

An NFSM $M$ is observable if and only if the corresponding FA $F(M)$ is deterministic. Clearly every DFSM is observable (the converse is not true). Any (completely specified) NFSM $M$ is equivalent to some observable NFSM (*ONFSM*). Observability will be discussed further in Section 3.

An NFSM is said to be *connected* if for every ordered pair of states $(s, s')$ there is some input sequence $x$ such that $s' \in h_1^*(s,x)$. Two states $s$ and $s'$ are said to be *quasi-equivalent* ($s =_q s'$) if for every input sequence $x$, $h_1^*(s,x) = h_1^*(s',x)$. An NFSM $M$ is said to be *reduced* if $M$ is connected



BURA

and no two states of $M$ are quasi-equivalent.

It is assumed throughout this paper that an implementation that behaves like some unknown DFSM $M_I$ is being tested against some NFSM $M$ that is reduced, connected, observable, and completely specified. Related work is discussed in Sections 2 and 3. In Section 4 the special case, when it is known that $M_I$ conforms to $M$ if and only if $M_I$ is a submachine of $M$, is discussed and the results are generalized in Section 5. The use of adaptive processes is explored in Section 6 and finally conclusions are drawn.

## 2 Testing From a DFSM

Throughout this section it will be assumed that $M$ is a reduced completely specified DFSM $(S, s_1, \delta, \lambda, X, Y)$ and $I$ behaves like some unknown reduced DFSM $M_I = (S', s'_1, \delta_1, \lambda_1, X, Y)$. As $M$ is deterministic, $I$ conforms to $M$ if and only if $L(M) = L(M_I)$. Thus, if $I$ conforms to $M$, there is a one-to-one correspondence between the states of $M$ and the states of $M_I$: $M$ and $M_I$ are isomorphic.

An input sequence $x$ is said to *distinguish* two states $s$ and $s'$ of $M$ if $\lambda^*(s, x) \neq \lambda^*(s', x)$. Similarly, an input sequence $x$ *distinguishes* between $M$ and $M_I$ if $\lambda^*(s_1, x) \neq \lambda_1^*(s'_1, x)$. The states $s$ and $s'$ of $M$ are *distinguishable* if there is some input sequence that distinguishes them. As $M$ is reduced, each pair of states from $M$ is distinguishable. A set of sequences $W$ is said to *verify* a state $s$ of $M$ if for each state $s' \in S$, $s \neq s'$, there is some $w \in W$ that distinguishes $s$ and $s'$. The set $W$ is said to be a *characterizing set* for $M$ if it verifies every state of $M$.

While a characterizing set is sufficient for state verification, two alternative approaches are commonly applied: using a distinguishing sequence or a set of unique input/output sequences (Sidhu and Leung [1988]). A *distinguishing sequence* for $M$ is an input sequence $D$ that verifies each state of $M$. Thus, if $M$ has a distinguishing sequence $D$ then $\{D\}$ is a characterizing set for $M$. A *unique input/output sequence* for a state $s$ of $M$ is an input sequence $x$ that verifies $s$. While $x$ can verify the state $s$ of $M$, it need not be able to verify any other state of $M$.

Unfortunately, not every DFSM has either a distinguishing sequence or a unique input/output sequence for every state. Every completely specified reduced DFSM has a characterizing set (Chow [1978]).

Chow [1978] introduces the assumption that there is some known upper



BURA

bound $m$ on the number of states of $M_I$. A *state cover* $V$ is a tree in which each node corresponds to a state of $M$, the branches correspond to transitions in $M$, and every state of $M$ has some corresponding node in $V$. Given a set $A$ and a natural number $k$, let $A^k$ denote the set of strings, from $A^*$, that have length at most $k$ (note, this is not the standard definition of $A^k$). If $V$ is a state cover and $W$ a characterizing set for $M$ then the test set $VX^{m-n+1}W$ will distinguish between $M$ and any nonconforming DFSM with no more than $m$ states: this test is called a *checking experiment*.

A sequence of transitions from $M$ is a *transition test* for a transition $t$ from $M$ if it consists of $t$ followed by either a unique input/output sequence for the final state of $t$ or a distinguishing sequence for $M$. A number of authors (e.g. Chow [1978], Aho et al. [1988], Yang and Ural [1990], Hierons [1996], Ural et al. [1997], and Hierons [1997b]) consider the case where $M$ is deterministic and $m = n$. They produce a single test sequence that contains, for each transition $t$ of $M$, a transition test. Most work is then based on trying to find the shortest test sequence with this property (e.g. Aho et al. [1988], Yang and Ural [1990], Hierons [1996], and Hierons [1997b]). Ural et al. [1997] instead assume there is a distinguishing sequence and find the shortest sequence that contains both a transition test, for each transition of $M$, and a set of subsequences that check the distinguishing sequence. This test is guaranteed to determine whether $M_I$ conforms to $M$, as long as $M_I$ has no more states than $M$, and is called a *checking sequence*.

# 3 Testing from an NFSM

## 3.1 Preliminaries

A number of problems are associated with testing nondeterministic systems. One problem is that it is impossible to be certain whether every possible response to a particular input sequence has been observed. Luo et al. [1994] add the test hypothesis, called the *complete testing assumption*, that there is some integer $\alpha$ such that if an input sequence $x$ has been executed $\alpha$ times from some state $s$ then it is guaranteed that every element of $h_2^*(s, x)$ has been produced. Thus, if a test set contains a number of input sequences, testing involves executing each $\alpha$ times.

Another problem is that, having executed an input sequence and observed the output sequence, there may be more than one current valid state for $M$.



BURA

A number of authors (e.g. Luo et al. [1994], Petrenko et al. [1994]) add the assumption that $M$ is observable, and note that any unobservable NFSM can be converted into an equivalent observable NFSM. The advantage of observability is that, although $M$ may be nondeterministic, the output of a transition defines the next state and thus eliminates one form of uncertainty. Thus, if an input sequence is executed from some state, by observing the output sequence the expected final state can be determined. As noted, it will be assumed that any NFSM considered is observable.

In testing it is common to assume that $M$ has *reset capability*: there is some special input that will always move $M$ to the initial state $s_1$. This allows a set of input sequences to be executed: they are simply separated by resets. Throughout this paper it will be assumed that $M_I$ has a *reliable reset*.

When considering NFSMs it is necessary to produce a new definition for a characterizing set. Petrenko et al. [1994] say that an input sequence $x$ *r-distinguishes* states $s$ and $s'$ if $h_2^*(s,x) \cap h_2^*(s',x) = \emptyset$. Thus, in any implementation that conforms to $M$, $x$ is guaranteed to distinguish between states that correspond to $s$ and $s'$. If $h_2^*(s,x) \neq h_2^*(s',x)$ but $h_2^*(s,x) \cap h_2^*(s',x) \neq \emptyset$ then the execution of $x$, from two states of $M_I$ corresponding to $s$ and $s'$, may lead to the same output.

States $s$ and $s'$ being r-distinguishable shall be denoted $s \neq_r s'$, otherwise $s =_r s'$. An NFSM is said to be *r-reduced* if for every $s$,$s' \in S$, $s \neq s' \bullet s \neq_r s'$. A set of sequences $W$ is a *characterizing set* if for all $s$,$s' \in S$, $s \neq_r s'$, there is some input sequence in $W$ that r-distinguishes $s$ and $s'$. It should be noted that it is only necessary to distinguish between states that are pairwise r-distinguishable. An NFSM that is not r-reduced is said to be *r-unreduced*.

Unfortunately the properties of being observable and r-reduced can conflict since, while any r-reduced unobservable NFSM $M$ can be converted into an ONFSM $M'$, $M'$ may not be r-reduced. This is the case in the example given in Figure 2. In fact, the following result shows that the states in $M'$ correspond to sets of states in $M$: if any pair of these sets intersect (and $M$ is completely specified) then $M'$ is r-unreduced. The machine given in Figure 2 is an example of this: the initial state of $M'$ corresponds to $\{s_1\}$, $u_2$ corresponds to $\{s_1, s_2\}$ and $u_3$ corresponds to $\{s_2\}$.

**Lemma 1** *If $M$ is completely specified, strongly connected, r-reduced, and unobservable and $M'$ is a strongly connected ONFSM that is equivalent to M then each state of $M'$ corresponds to a set of states from $M$.*



BURA

Proof

Let $A_s$ denote the set of input/output sequences that move $M'$ from its initial state to state $s$. Let $S_{A_s}$ denote the set of final states allowed after the input/output sequences from $A_s$ are executed from the initial state of $M$. As $M'$ is observable, the input/output sequences in $A_s$ have no other possible final state in $M'$. Thus the set of input/output sequences executable from $s$ must correspond to the union of the sets of input/output sequences allowed from the states in $S_{A_s}$. Thus $s$ corresponds to the state set $S_{A_s}$. □

This result suggests that examples such as that given in Figure 2 may be common. In particular, if $M'$ has more states than $M$ there must be some intersection and thus $M'$ will be r-unreduced. The example given in Figure 3 shows, however, that it is possible for $M'$ to be r-reduced.

## 3.2 Test Generation

This section will describe results and algorithms, found in Yevtushenko et al. [1991] and Petrenko et al. [1994], for generating a checking experiment from a reduced ONFSM. An initial result given is that if $M$ has $n$ states and $M_I$ has no more than $m$ states then the input sequence $X^{mn}$ is a checking sequence.

Both then look at conditions that allow this set to be reduced. They assume that there is a set $V$ such that for each state $s_i$ there is some input sequence $v_i \in V$ with the property that $h_2^*(s_1, v_i) = \{s_i\}$: the input sequence $v_i$ is guaranteed to bring the NFSM to state $s_i$. The set $V$ is called a *deterministic state cover*. The NFSM $M_0$ has, for example, a deterministic state cover $V = \{\epsilon, b, ba, baa, baaa\}$. They produce a test technique, in the presence of a deterministic state cover, that utilizes states being r-distinguishable, but does not require the NFSM to be r-reduced. For each state $s_i \in S$, the set $W_i$ denotes the set of input sequences used to distinguish between $s_i$ and each $s_j$ such that $s_j \neq_r s_i$.

Let $u_1$ denote the initial state of $M_I$. Yevtushenko et al. [1991] and Petrenko et al. [1994] consider a tree with root $(s_1, u_1)$ and edges corresponding to input/output pairs that are allowed by both $M$ and $M_I$. Then a node is a leaf if one of the following is the case:

1. The state pair has already appeared somewhere else in the tree as an intermediate node.



BURA

2. There is some input value such that $M$ and $M_I$ do not have matching transitions.

Then $M_I \leq M$ if and only if all the leaves are of type 1.

Let $P_1, \ldots, P_k$ denote the maximal sets of r-distinguished states from $M$ and, for each $s_i \in S_M$, $Q(s_i)$ denote the set of states from $S_{M_I}$ that agree with $s_i$ on $W_i$. If $M$ is r-reduced and $M_I \leq M$, there is only one such set: $P_1 = S_M$. By considering the possible pairs and using the fact that each state $s_i$ is in some pair in $V$, the following result is obtained:

- if the states from some $P_j$ are met $\sum_{s_i \in P_j}(|Q(s_i)| - 1) + 1$ times in some path after a sequence from $V$ then a leaf must have been met.

Thus it is sufficient to stop a path when there is some $P_j$ such that the path contains this number of instances of states from $P_j$. While the $Q(s_i)$ are not known, this expression is bounded above by $m - |P_j| + 1$.

The maximal sets of pairwise r-distinguishable states, $P_1, \ldots, P_k$, are found and the test set is generated in the following manner (Yevtushenko et al. [1991], Petrenko et al. [1994]):

1. For each state $s_i$ of $M$ let $v_i$ denote the input sequence in $V$ that reaches $s_i$. For each state $s_i$ a tree $D_i$, starting with $s_i$, is constructed. The nodes represent states of $M$, while the edges represent possible transitions. A node is a leaf if there is some $P_j$ such that the states from $P_j$ are met $(m - |P_j| + 1)$ times in total on the path to that node (not counting the root node).

2. Then, for an input/output sequence $x/y$ in $D_i$ with final state $s_l$, the test $v_i x W_l$ is included. The empty sequence is included in the set of sequences from $D_i$.

3. The test set is the set of all such input/output sequences.

Clearly the size of this test set depends on the number of states that are pairwise r-distinguished: as the number of r-distinguished states reduces, the size of the $P_j$ reduces and thus the size of the test set increases. In Sections 4 and 5 $\neq_r$ will be extended, thus potentially increasing the size of the $P_j$ and thus reducing the size of the checking experiment produced.

# 4 Testing Deterministic Submachines

## 4.1 Deterministic Equivalence

An NFSM $M'$ is a *submachine* of $M$ if it is isomorphic to some NFSM $M_S$ whose state set and transition set are subsets of those of $M$. It will be assumed throughout this section that if $M_I \leq M$ then $M_I$ is a submachine of $M$. This condition will be weakened in Section 5.

The concept of state distinguishing can be extended in this case. A set $A$ of input sequences is said to *d-distinguish* states $s$ and $s'$ of $M$ if for every deterministic submachine $M'$ of $M$, and corresponding states $u$ and $u'$ from $M'$, there is some $x \in A$ that distinguishes $u$ and $u'$. If $s$ and $s'$ are d-distinguishable we write $s \neq_d s'$ and otherwise $s =_d s'$.

In $M_0$ there are states that are pairwise d-distinguishable but not r-distinguishable. The input sequence $aaa$ will, for example, distinguish between the states corresponding to $s_1$ and $s_2$ in any deterministic submachine: from the state corresponding to $s_2$ it will produce output 001 while from the state corresponding to $s_1$ it will either produce 000 or 110.

In the unobservable case described in Figure 4, two input sequences are required: $aa$ and $ab$. Each choice for the execution of $a$ from $s_1$ will lead to a following transition that is not allowed after the execution of $a$ from $s_2$.

The algorithms given in Petrenko et al. [1994], that use $=_r$, can be applied using $=_d$ instead. As the use of $=_d$ can increase the size of the sets of pairwise distinguished states, it can reduce the size of the test set. Clearly, $s_i \neq_r s_j \Rightarrow s_i \neq_d s_j$, and thus d-distinguishability can never lead to longer tests that r-distinguishability.

## 4.2 Finding d-distinguishing sets

Suppose $s_i =_r s_j$ but $s_i$ and $s_j$ are not quasi-equivalent. Then there may be some set of input sequences that d-distinguished $s_i$ and $s_j$. The obvious approach, to finding a d-distinguishing set, is to use a breadth first search of a tree starting with $(s_i, s_j)$. In this tree, edges represent input values. Each node represents the possible configurations, given the input so far, that are consistent with the same output having been produced from each state. Thus a node represents a set of tuples, where each tuple contains the corresponding states $s_i'$ and $s_j'$ and the deterministic choices that are required in order to allow $M_I$ to move from $s_i$ to $s_i'$ and from $s_j$ to $s_j'$ producing the same output sequence. Given input sequence $x$, let $c(x)$ denote the set of tuples corresponding to $x$ and $\pi_k$ denote the projection function that returns the $k$th element of a tuple.

A set of choices can be represented by a predicate $p$, which takes a deterministic submachine $M'$ of $M$ and returns true if and only if $M'$ allows those choices. Thus a node contains a set of tuples of the form $(s'_i, s'_j, p)$. The form of $p$ depends upon the representation of the choices.

What is required is one of:

1. Some input sequence $x$ such that $c(x) = \emptyset$.
2. Some set of input sequences $x_1, \ldots x_r$ such that for every $p_1, \ldots p_r$, $p_q \in \pi_3(c_q)$ for some $c_q \in c(x_q)$, and every deterministic submachine $M'$ of $M$, $\neg(p_1(M') \wedge p_2(M') \wedge \ldots \wedge p_r(M'))$.

In the first case, only one sequence is required, in the second a set of sequences is required. For pragmatic reasons, limits can be placed on the size of sets and sequences considered in the search: only those that can reduce the test effort are of interest.

# 5 Deterministic implementations that are not submachines

## 5.1 Extending deterministic distinguishing

Let the states of $M$ be denoted $s_1, \ldots, s_n$ and let $V$ denote a deterministic state cover of $M$. In order to generalize the notion of deterministically distinguishing states it is sufficient to consider all deterministic reductions of $M$ that have no more than $k$ states that cannot be reached by $V$. Given two states, $s_i$ and $s_j$, we require a set of input sequences that is guaranteed to distinguish between any pair of corresponding states in the implementation.

Let $L(s)$ denote the set of input/output sequences allowed from the state $s$ and $s \leq s'$ denote $L(s) \subseteq L(s')$. It is important to note that if $M_I \leq M$ then, for any reachable state $s \in S_{M_I}$, there is some $s' \in S_M$ such that $s \leq s'$.

We say that states $s_i$ and $s_j$ are *deterministically* $(V, k)$ *distinguished* if there is a set of input sequences $A$ such that, for every DFSM $M' = (U, u_1, \delta, \lambda, X, Y)$ that conforms to $M$ and has no more than $k$ states that are not reached by $V$:



BURA

- given $u_i$, $u_j \in U$ with $u_i \leq s_i$ and $u_j \leq s_j$ there is some input sequence $x \in A$ that distinguishes between $u_i$ and $u_j$.

This is written $s_i \neq_{(V,k)} s_j$ and otherwise $s_i =_{(V,k)} s_j$. An NFSM $M$ is said to be *deterministically* $(V, k)$ *reduced* if for every $s_i$,$s_j \in S_M$, $s_i \neq_{(V,k)} s_j$ and otherwise $M$ is said to be *deterministically* $(V, k)$ *unreduced*. Clearly, $=_{(V,0)}$ is equivalent to $=_d$ and as $k \to \infty$, $=_{(V,k)} \to =_r$.

The NFSM $M_0$ given in Figure 1 has a deterministic state cover $V = \{\epsilon$, $b$, $ba$, $baa$, $baaa\}$, where $\epsilon$ denotes the empty sequence. Consider the input of $aaa$. If $M_I$ is in a state $s \leq s_2$, the input of $aaa$ leads to output 001. If $k = 0$ and $aaa$ is input, while $M_I$ is in a state $s \leq s_1$, either 000 or 110 can be is output. If $k = 1$ and $a$ is input while $M_I$ is in a state $s \leq s_1$, $M_I$ might output 0 and move to another (non-equivalent) state $s' \leq s_1$. The input of $aaa$, from $s$, then leads to the output of 011. Thus if $aaa$ is input and $M_I$ is in state $s$, if $s \leq s_2$ then $M_I$ outputs 001 and if $s \leq s_1$ and $k = 1$ then the output sequence generated is one of: 000, 110, or 011. Thus, $s_1$ and $s_2$ are deterministically $(V, 1)$ distinguished by $aaa$. It is easy to check that $s_1 =_{(V,1)} s_5$, $s_1 =_{(V,1)} s_4$ and all other state pairs are r-distinguished.

## 5.2 Finding the value of $k$

If there is some upper bound $m$ on the number of states of $M_I$, an upper bound can be found for $k$. Suppose $S'$ denotes a maximal (in terms of size) set of pairwise r-distinguished states of $M$. Then, for any conforming implementation, each of these must have a corresponding separate state reached by $V$. Thus, an initial upper bound of $k = m - |S'|$ can be used. There may be further information, about the implementation, that can be used to reduce this.

The NFSM given in Figure 1 has maximal sets of pairwise r-distinguished states $P_1 = \{s_2, s_3, s_4, s_5\}$ and $P_2 = \{s_1, s_3\}$. If the value $m = n = 5$ is used, an upper bound of $k = 1$ is found. This information helps reduce the required test size as $s_1$ and $s_2$ are deterministically $(V, 1)$ distinguished and thus the set $P_2$ can be extended to $P_2' = \{s_1, s_2, s_3\}$. The set $P_1$ is not affected and thus $P_1' = P_1$.

In this case, the tree derived from each state has the property that each leaf represents meeting the states from $P_1'$ $m - |P_1'| + 1 = 2$ times or meeting the states from $P_2'$ $m - |P_2'| + 1 = 3$ times. The characterizing set is $W = \{aaa\}$. Input $aaa$ produces output 000 or 011 or 110 from $s_1$, 001 from $s_2$,



BURA

101 from $s_3$, 011 from $s_4$, and 110 from $s_5$. This fails to distinguish $s_1$ from either $s_4$ or $s_5$.

It is now possible to derive a test set. In this case every input sequence of length 2, from any state, will pass through the elements of some $P'_i$ twice. As an example, we will consider the input sequence $bb$. From $s_1$ this will reach states $s_2$ and then $s_1$, both of which are from $P'_2$. Similarly, from $s_2$ it goes to $s_1$ and then $s_2$. From $s_3$ it moves to $s_4$ and then $s_5$ or $s_2$; in each case both states are in $P'_1$. From $s_4$ it passes through $s_2$ or $s_5$ to $s_1$ or $s_2$ respectively. In the first case both are in $P'_2$ and in the second both are in $P'_1$. Finally, from $s_5$ it will reach $s_2$ and then $s_1$ which are both in $P'_2$. The checking experiment thus contains the following:

$$\{\epsilon, b, ba, baa, baaa\}\{\epsilon, a, b, aa, ab, ba, bb\}\{aaa\}.$$

It is also necessary to consider sequences of length 3 from $M_0$. Any sequence of length 3, such that the first two states it meets need not be from $P'_1$, must be included. An example is the sequence $aba$ from $s_1$: this might go to $s_1$, then $s_2$ and finally $s_4$. It is easy to check that no sequences of length greater than 3 need be considered: every sequence of length 3 in $M_0$ meets either at least two states from $P'_1$ or three states from $P'_2$. The sequences of length 3 that must be included (and thus, in the checking experiment, preceded by the corresponding $v_i$ and followed by $W$) are:

1. From $s_1$: any string starting with either $a$ or $bb$.
2. From $s_2$: any string starting with $b$.
3. From $s_3$: none.
4. From $s_4$: any string starting with $bb$.
5. From $s_5$: any string starting with $bb$.

The checking sequence generated can be reduced by removing those sequences that are contained in the beginning of other sequences.



BURA

# 6 Adaptive Testing

## 6.1 The deterministic state cover

When input sequences are applied to $M_I$, the output provides information about $M_I$. Clearly, $M_I$ has some deterministic state cover $V$. This can be developed by using a breadth first search, at each step simply executing candidate values. The output determines the (expected) next state and thus whether the expected next state is one already included in the tree. Given an input sequence $x$ that provides a new expected state, and thus will be used in $V$, at this point $x$ can be followed by the appropriate $W_i$.

There are a number of possible orders in which to execute the required sequences. One possibility is to initially execute $W$ (or the corresponding set required for $s_1$). This provides information, about $M_I$, which can be used to provide part of $V$. The rest of $V$ is developed in an adaptive manner. This search is continued until a deterministic state cover $V$ has been found.

## 6.2 Testing Submachines

Suppose the set $W$ has been produced for any deterministic FSM that is a submachine of $M$. The particular IUT, $M_I$, may have properties that mean that not all of these sequences are required. If two states $s_i$ and $s_j$ are deterministically distinguished by a set $A$ then in $M_I$ one input sequence $\alpha \in A$ will distinguish between the corresponding states. Once $\alpha$ has been found, $A$ can be replaced by $\alpha$, or some initial subsequence of $\alpha$, in $W$. Similarly, some other sequences in $W$ may not be required when testing $M_I$, and thus may be removed, and others may be shortened. Thus, once $VW$ has been executed, each $W_i$ can be replaced by some subset of (possibly shortened) sequences drawn from $W_i$.

## 6.3 General deterministic implementations

While it is still always possible to devise a state cover $V$ for $M$ that is implemented (and deterministic) in $M_I$, this may not reach all states in $M_I$. If some states of $M_I$ may not be reached by executing $V$, it is not possible to reduce the size of $W$ using information derived from the execution of $VW$.

As noted in Section 5, the value of $k$ can be derived from an upper bound $m$ on the number of states of $M_I$. Associated with $W$ and $M$ there is some



BURA

maximal (in terms of size) set of states $S'$ that are pairwise distinguished by $W$, and thus a value of $k = m - |S'|$ can be used. The set $S'$ is based on states that are guaranteed to be distinguished by $W$. Once $VW$ has been executed, it may transpire that there is some larger set of states $S''$ reached by $V$ that is pairwise distinguished, in $M_I$, by elements from $W$. The value $k = m - |S''|$ can then be used and this reduction in the value of $k$ can potentially further increase the set of pairwise deterministically $(V, k)$ distinguished states.

The knowledge, of the behaviour of certain instances of states from $S$, can also be used to directly reduce the size of the test sequence. This is because, given some $P_j$, there may be some state in $M_I$ that can be used to extend $P_j$. This happens if there is an input/output sequence $x/y$ whose final state in $M$ is $s_i$ for some $s_i \notin P_j$ but the corresponding state $u$ in $M_I$ has the property that for all $s \in P_j$, $u \notin Q(s)$. Thus it may be possible to extend a set $P_j$ by some maximal (size) set $\overline{P}_j$ of states from $M_I$ such that the states in $\overline{P}_j$ are pairwise distinguishable and each is pairwise distinguishable from each state in $P_j$. Then:

$$|\overline{P}_j| + \sum_{s_i \in P_j} |Q(s_i)| \leq m$$

Given $P_j$, this information gives the following upper bound on the number of occurrences of states from $P_j$ in a test sequence:

$$m - |P_j| - |\overline{P}_j| + 1$$

As testing proceeds, these values can be updated.

An input sequence $x$ can be seen as a route to some state of $M_I$. It is possible to update these values by considering the execution of $W$ at the end of routes. As each sequence from $D_i$ is followed by some $W_l$, this fits the test technique.

Suppose that, in the example, $W = \{aaa\}$ is initially executed from $s_1$. If this were to produce output 000 then this instance of $s_1$ would be distinguishable from both $s_4$ and $s_5$. Thus $s_4$ and $s_5$ can be included in $\overline{P}_2$ and $s_1$ can be included in $\overline{P}_1$. Further, as $m = 5$ and $M_I$ is reduced, the set $\{aaa\}$ must be a characterizing set for $M_I$. Thus the test set can be reduced to:

$$\{\epsilon, b, ba, baa, baaa\}\{\epsilon, a, b\}\{aaa\}$$

Again, sequences contained in the beginning of others can be removed.



BURA

# 7 Conclusions

The problem of testing a nondeterministic implementation against an NFSM has received much attention but there has been little work on testing from an NFSM when the IUT is known to be deterministic. When the IUT is deterministic, it is possible to generalize the notion of r-distinguishing states. When it is known that if the IUT conforms to $M$ then it is a submachine of $M$, it is possible to d-distinguish states. When instead there is some upper bound $k$ on the number of states in the IUT that are not reached by the deterministic state cover $V$, it is possible to consider $(V, k)$ distinguishing states. In the case where $k = 0$, this reduces to d-distinguishing states.

When the IUT is deterministic, much can be learnt about the structure of the IUT during test execution. A test can thus be generated in an adaptive manner. This guarantees the existence of a deterministic state cover and allows the test set to be reduced as testing proceeds.

An interesting question, when applying adaptive testing, is how the test order than maximizes the expected reductions can be found. There is also the problem of limiting the search for deterministically $(V, k)$ distinguishing sets. There may be no good upper bound on the size of these: instead limits can be placed on the size of sets that could reduce the total test effort.

# 8 References


1. A.V. Aho, A.T. Dahbura, D. Lee, and M.U. Uyar, 1988, An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours *Proceedings of Protocol Specification, Testing, and Verification VIII*, pp.75-86, Atlantic City, North-Holland.

2. T.S. Chow, 1978, Testing Software Design Modelled by Finite State Machines, *IEEE Transactions on Software Engineering*, **4** 3, March 1978, pp.178-187.

3. J. Dick and A. Faivre, 1993, Automating the Generation and Sequencing of Test Cases from Model-Based Specifications, *FME '93, First International Symposium on Formal Methods in Europe*, Odense, Denmark, 19-23 April 1993, pp.268-84.




BURA

4. A.Gill, 1962, *Introduction to The Theory of Finite State Machines*, McGraw-Hill.

5. R.M. Hierons, 1996, Extending Test Sequence Overlap by Invertibility, *The Computer Journal*, **39** 4, pp.325-330.

6. R.M. Hierons, 1997a, Testing from a Z Specification, *Journal of Software Testing, Verification, and Reliability*, **7** 1, pp.19-33.

7. R.M. Hierons, 1997b, Testing from a Finite State Machine: Extending Invertibility to Sequences, *The Computer Journal*, **40** 4, pp.220-230.

8. Z. Kohavi, 1978, *Switching and Finite State Automata Theory*, McGraw-Hill.

9. G. Luo, G. v. Bochmann, and A. Petrenko, 1994, Test Selection Based on Communicating Nondeterministic Finite-State Machines Using a Generalized Wp-Method, *IEEE Transactions on Software Engineering*, **20** 2, pp.149-161.

10. A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das, 1994, Nondeterministic State Machines in Protocol Conformance Testing, *Proceedings of Protocol Test Systems, VI (C-19)*, pp.363-378.

11. M.O. Rabin and D. Scott, 1959, Finite Automata and Their Decision Problems, *IBM Journal of Research and Development*, **3** 2, pp.114-125.

12. D. Sidhu and T. K. Leung, 1988, Experience with Test Generation for Real Protocols, *ACM SIGCOMM 88*, pp257-261.

13. A.S. Tanenbaum, 1996, *Computer Networks*, Prentice Hall, International Editions, 3rd edn.

14. H. Ural, X. Wu, and F. Zhang, 1997, On Minimizing the Lengths of Checking Sequences, *IEEE Transactions on Computers*, **46** 1, pp.93-99.

15. B. Yang and H. Ural, 1990, Protocol Conformance Test Generation Using Multiple UIO Sequences with Overlapping, *ACM SIGCOMM 90: Communications, Architectures, and Protocols*, Sept 24-27 pp.118-125, Twente, Netherlands, North-Holland.



BURA

16. N.V. Yevtushenko, A.V. Lebedev, and A.F. Petrenko, 1991, On Checking Experiments With Nondeterministic Automata, *Automatic Control and Computer Sciences*, **6**, pp.81-85.