Generating Candidates when testing a deterministic implementation against a Non-deterministic Finite State Machine

R. M. Hierons, Brunel University, Uxbridge, Middlesex, United Kingdom

October 7, 2004

Abstract

This paper considers the problem of testing a deterministic system against a non-deterministic finite state machine. An adaptive test process, with two phases, is proposed. The paper focuses on the first stage which involves testing to generate a *candidate* deterministic finite state machine. This candidate has the property that, under the test hypotheses used, the implementation is correct if and only if it is equivalent to the candidate. A test may then be derived from the candidate.

keywords: non-deterministic finite state machine, deterministic implementation, conformance testing, test derivation[1].

1 Introduction

A number of classes of system, such as control systems [11] and communication protocols [18], may be described using non-deterministic finite state machines. While non-determinism aids abstraction, and thus is highly appropriate for specifications, actual implementations are typically deterministic. Given an implementation I and a non-deterministic finite state machine (NFSM) M that models the required behaviour of I, it is important to check I against M. Testing usually forms part of this process.

Finite state machine (FSM) based test techniques are often applied when testing from a specification, in a language such as SDL [10] or Statecharts [6], that is an extended finite state machine (EFSM). Typically the EFSM is converted into an FSM, by either applying an abstraction or expanding out the data once the types have been made finite. Tests are generated from the resultant FSM.

Most approaches to testing from an NFSM produce a preset test set [13, 14, 15]. The test is typically associated with a set of assumptions about the *implementation under test (IUT)* I, called *test hypotheses*. During testing, however, the response of I to input sequences provides information about I. Where the implementation is known to behave like some (unknown) deterministic finite state machine (DFSM) this information can be used to reduce the test effort. In particular, it may be used in order to direct further testing: *adaptive testing* may be applied [8].

This paper considers the problem of testing a deterministic implementation, that behaves like some unknown DFSM with at most m states, against NFSM M. The paper introduces a new approach, to adaptive testing, that involves two stages. In the first stage, an adaptive test procedure is applied in order to generate a candidate M^C. The candidate is a DFSM with the property that, assuming I satisfies the test hypotheses, I conforms to M if and only if I is equivalent to M^C. Tests may then be generated from M^C.

This approach adapts itself to the IUT. This contrasts with the usual test methods. The test generated from M^C may thus be significantly smaller than that generated from M. This process also has the advantages that it is easier to test from a DFSM than from an NFSM and there are many techniques for generating tests from a DFSM.

This paper focuses on the problem of generating a candidate DFSM where the implementation is deterministic and the specification is an NFSM. It defines provably correct algorithms that achieve this. Section 2 introduces a number of notions and definitions while Section 3 briefly reviews related work. Section 4 considers the case where M is r-reduced. Section 5 extends this to the general case. In Section 6 the method outlined in

[1]This is a preliminary version of the following paper: R.M. Hierons, 2003, Using Candidates to test a Deterministic Implementation against a Non-deterministic Finite State Machine, *The Computer Journal*, **46** 3, pp. 307-318.

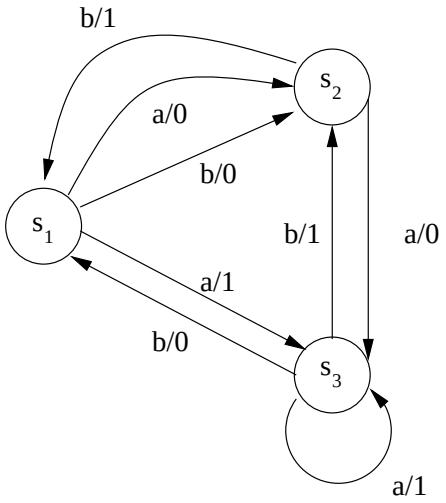

Figure 1: The NFSM M_0

Section 5 is compared with a previous approach to testing a deterministic implementation against an NFSM. Section 7 then makes a number of observations and discusses possible future work. Finally, conclusions are drawn.

2 Finite State Machines

A *finite automaton* (*FA*) N is defined by a tuple $(S, s_1, \gamma, \Psi, F)$, in which S is a finite set of *states*, $s_1 \in S$ is the *initial state*, γ is the (possibly partial) *state transition function*, Ψ is the finite *alphabet* and $F \subseteq S$ is the set of *final states*. The function γ takes a state $s \in S$ and a value $x \in \Psi$ and returns a set of possible next states $\gamma(s, x)$. Let Ψ^* denote the set of strings consisting of elements from Ψ, including the empty sequence ϵ. The transition function γ can be extended to take values from Ψ^*, giving the function γ^*.

A string $x \in \Psi^*$ is *accepted* by N if and only if $\gamma^*(s_1, x) \cap F \neq \emptyset$: x *can* take N to a final state. The FA N defines a language $L(N)$, which is the set of strings accepted by N, and is deterministic if for each $s \in S$ and $x \in \Psi$, $|\gamma(s, x)| \leq 1$. Given a non-deterministic FA it is possible to generate a deterministic FA that defines the same language [16].

A (completely specified) finite state machine is essentially a finite automaton in which each transition has an associated output value. In this paper only completely specified finite state machines will be considered and thus all definitions and results will refer to completely specified finite state machines. A non-deterministic finite state machine M is defined by a tuple $(S, s_1, h, \Sigma, \Gamma)$ in which S is a finite set of *states*, s_1 is the *initial state*, Σ is the finite *input alphabet*, Γ is the finite *output alphabet* and h is the *state transition function*. When M receives input $x \in \Sigma$, while in state $s \in S$, a *transition* is executed producing an output value $y \in \Gamma$ and moving M to some state $s' \in S$. These satisfy $(s', y) \in h(s, x)$ and thus h gives the possible transitions. An NFSM, that will be denoted M_0 throughout this paper, is given in Figure 1. Here, for example, $h(s_1, a) = \{(s_2, 0), (s_3, 1)\}$.

An NFSM M can be represented by a FA $\mathcal{F}(M)$ in which Ψ is the set of input/output pairs and $F = S$. A state s of $\mathcal{F}(M)$ is characterized by the set $L_M(s)$ of input/output sequences that M allows from s. Similarly, M is characterized by $L_M(s_1)$, which will be denoted $L(M)$.

Next state function h_1 and output function h_2 can be defined in terms of h: if $(s', y) \in h(s, x)$ then $s' \in h_1(s, x)$ and $y \in h_2(s, x)$. The functions h, h_1, and h_2 can be extended, in a similar manner to γ, to take a state and an input sequence giving functions h^*, h_1^*, and h_2^* respectively.

Suppose $t = (s, s', x/y)$ is a transition of NFSM M. Then t is a *deterministic transition* if M has no other transition from state s with input x. Otherwise t is a *non-deterministic transition*.

An NFSM M is *deterministic* if for each $s \in S$ and $x \in \Sigma$, $|h(s, x)| \leq 1$. If M is deterministic the transitions are defined by a *next state function* $\delta = h_1$ and an *output function* $\lambda = h_2$. A deterministic finite state machine can thus be defined by the tuple $(S, s_1, \delta, \lambda, \Sigma, \Gamma)$. Again, δ and λ can be extended to take input sequences,

giving functions δ^* and λ^* respectively.

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

NFSM M' *conforms* to NFSM M if $L(M') \subseteq L(M)$ and this is denoted $M' \leq M$ [14]. This notation may be extended to states: given states s (of M) and s' (of M'), $s' \leq s$ if and only if $L_{M'}(s') \subseteq L_M(s)$. If $s' \leq s$, s' *conforms* to s. Input/output behaviour x/y is *consistent* with M if and only if $x/y \in L(M)$.

For each $y \in \Gamma$, $s \in S$, and $x \in \Sigma$, $h_y(s,x)$ will denote the set of possible next states if a transition executed from s, using input x, produces output y. This can be extended to h_y^* (here y is an output sequence). An NFSM M is said to be *observable* if for every $s \in S$, $x \in \Sigma$, and $y \in \Gamma$, $|h_y(s,x)| \leq 1$ [17]. 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: if the output from the input of a in the initial state is 0 then the state reached is s_2 and otherwise it is s_3. This is a general property of observable NFSMs which reduces the uncertainty caused by non-determinism.

An NFSM M is observable if and only if the corresponding FA $\mathcal{F}(M)$ is deterministic. Thus, as each non-deterministic FA is equivalent to some deterministic FA, each NFSM M is equivalent to some observable NFSM (*ONFSM*).

An NFSM is *initially connected* if every state can be reached from its initial state: for all $s \in S$ there exists some $x \in \Sigma^*$ such that $s \in h_1^*(s_1, x)$. An NFSM is *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 *equivalent* if for every input sequence x, $h_2^*(s,x) = h_2^*(s',x)$. An NFSM M is *reduced* if it is initially connected and no two states of M are equivalent. An ONFSM can always be rewritten to an equivalent reduced ONFSM.

Let Φ_M^m denote the set of reduced, initially connected DFSM with the same input and output alphabets as M and no more than m states. In this paper it will be assumed that some IUT I, which behaves like an unknown (completely specified) DFSM $M_I = (T, t_1, \delta_I, \lambda_I, \Sigma, \Gamma) \in \Phi_M^m$, is being tested against a completely specified reduced ONFSM $M = (S, s_1, h, \Sigma, \Gamma)$. During testing it is normal to use a reset operation in order to return I to its initial state. It will be assumed that I has a *reliable reset operation*: a reset operation that is known to be implemented correctly. Potentially this might involve the system being switched off and then switched on again.

3 Related Work

Testing I against M involves executing I with a number of input sequences and comparing the output produced with those allowed by M. An input sequence used during testing is called a *test sequence*. A set of test sequences is called a *test set* or a *test suite*. Once a test sequence has been executed and the output noted, the system is reset and is ready for the next test.

When testing from an NFSM it is normal to have some fault model [9] or a set of test hypotheses [4]. A test hypothesis is some property that the tester believes that the IUT has. A fault model is some set of faulty behaviours with the property that the tester believes that if the IUT is faulty then it behaves like some unknown element of this set. Naturally the two concepts are similar and in this paper the term test hypothesis will be used. Thus the test hypothesis is: the IUT behaves like an unknown completely specified DFSM $M_I \in \Phi_M^m$ that has a reliable reset.

This section will start with a number of definitions and then describe a related test generation algorithm.

3.1 Preliminaries

In testing it is often important to be able to distinguish states. Sometimes it is possible to produce a set of input sequences that, between them, distinguish two states of an NFSM M [15]. Two states s and s' are *r(1)-distinguishable* if there exists an input value $x \in \Sigma$ such that $h_2(s,x) \cap h_2(s',x) = \emptyset$. Further, s and s' are *r(i)-distinguishable* ($i > 1$) if there exists some j, $1 \leq j < i$ such that one of the following holds:

1. s and s' are r(j)-distinguishable;
2. there exists $x \in \Sigma$ such that for all $y \in h_2(s,x) \cap h_2(s',x)$, $h_y(s,x)$ and $h_y(s',x)$ are r(j)-distinguishable.

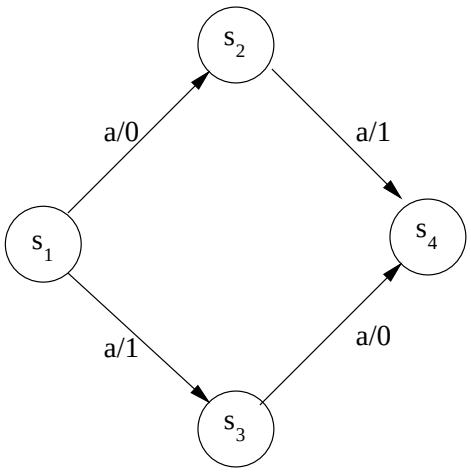

Figure 2: State s_4 is deterministically reachable by aa

If there exists some i such that s and s' are r(i)-distinguishable then s and s' are *r-distinguishable* [15]. This leads to the use of a finite set X of input sequences to *r-distinguish* s and s'. s and s' being r-distinguished by X is denoted $s \not\sim_X s'$; otherwise $s \sim_X s'$. If the states of an NFSM M are pairwise r-distinguishable, M is *r-reduced*.

A set W of input sequences is a *characterizing set* if for each pair of r-distinguishable states (s, s') of M, $s \not\sim_W s'$. M_0, for example, is r-reduced and has characterizing set $\{aa, bb\}$. Given a characterizing set W and state s, W_s will denote some minimal set of prefixes of input sequences from W such that: for every state s' that is r-distinguishable from s, $s \not\sim_{W_s} s'$. Often W_s is called a *state identifier*.

Given an NFSM M with n states and an implementation I that behaves like some unknown NFSM with at most m states, a (preset) set C of input sequences is a *checking experiment* if for every NFSM M_I with the same input alphabet as M and at most m states, $M_I \leq M$ if and only if $M \sim_C M_I$. There has been much interest in the generation of checking experiments from DFSMs and NFSMs [2, 7, 8, 14, 15, 19, 20]. However, the paper will propose an alternative, adaptive, approach. Note that some authors use the term checking experiment to simply mean a test suite or test set.

A state s of M is *deterministically reachable*, or *d-reachable*, if and only if there is some input sequence v that moves M from s_1 to s only: $h_1^*(s_1, v) = \{s\}$. Here v is said to *d-reach* s. If all of the states of M are deterministically reachable then M is said to be *d-connected* [14, 15]. A set V_{det} of input sequences is a *deterministic state cover* for M if the empty sequence ϵ is contained in V_{det} and for each deterministically reachable state s of M there is some $v \in V_{det}$ such that v takes M from its initial state to s only. As illustrated by Figure 2, the transitions within such a sequence need not all be deterministic.

Where M is a reduced DFSM, M has a characterizing set W that distinguishes all states of M. Then the set $V_{det}(\Sigma^0 \cup \Sigma^1 \cup \ldots \cup \Sigma^{m-n+1})W$ is a checking experiment [21, 2]. This approach, to generating a checking experiment, is often called the *W-method*.

3.2 Test generation

A number of authors have studied test generation where the specification is non-deterministic and the implementation is deterministic [1, 8, 15]. This case is important: while most implementations are deterministic the use of non-determinism aids abstraction and so is appropriate for specifications.

Throughout this section it will be assumed that M is a reduced ONFSM with n states and I satisfies the test hypotheses. Petrenko et al. [15] give a general method for generating a (preset) checking experiment under these conditions. This method will now be described.

Suppose $S_{V_{det}}$ denotes the set of d-reachable states of M and V_{det} denotes a deterministic state cover. Given state $s \in S_{V_{det}}$, let v_s denote the input sequence from V_{det} that d-reaches s. Suppose also that W denotes a characterizing set. Let $D_1, \ldots, D_p$ denote the maximal sets of pairwise r-distinguishable states of M and, given

D_i, $\widehat{D}_i$ denote the elements of D_i that are deterministically reachable: $\widehat{D}_i = D_i \cap S_{V_{det}}$.

Given a state s from $S_{V_{det}}$, a *traversal set* $Tr(s)$ for s is defined by [15]:

- $Tr(s)$ is the set of (minimal) $x \in \Sigma^*$ such that for every output sequence $y \in h_2^*(s,x)$ there is some D_i such that the sequence of transitions associated with x/y being observed from state s meets states from D_i at least $m - |\widehat{D}_i| + 1$ times.

Let *prefix*(x) denote the set of prefixes of the sequence x: *prefix*$(x) = \{x' | (\exists x''.x = x'x'')\}$. A checking experiment may be generated by the union, over states $s \in S_{V_{det}}$ and $x \in Tr(s)$, of the set of $\{v_s\}$*prefix*$(x)W$ [15]. The size of this test may be further reduced by using the W_s rather than W [15].

The test size is greatly affected by the size of the $|\widehat{D}_i|$ and thus by the size of the maximal sets of r-distinguishable states of M and the number of d-reachable states of M. In the ideal case, where M is d-connected and r-reduced, the test set is equivalent to that generated by the W-method.

Where I is deterministic, the notion of r-distinguishing may be extended [8]. Suppose there is some deterministic state cover V_{det} of M and at most k states of M_I are not reached by V_{det}. Two states s and s' of M are *deterministically* (V_{det}, k) *distinguishable* if in every DFSM that conforms to M and has at most k states not reached by V_{det}, no state conforms to both s and s'. Two states of M may be deterministically (V_{det}, k) distinguishable but not r-distinguishable [8]. By extending the notion of state distinguishing, shorter checking experiments may be generated.

In principle it may be possible to either generate all elements of Φ_M^m that conform to M or all possible combinations of non-deterministic choices. Test generation might then be based on this. This paper introduces algorithms that do not suffer from the storage issues of such approaches.

4 Testing when M is r-reduced

This section will consider the case where M is r-reduced. This will be generalized in Section 5. I will be tested in order to derive a candidate M^C with the property that, assuming the test hypotheses hold, I conforms to M if and only if M_I is equivalent to M^C. A test may then be generated from M^C. A breadth-first search of the states of M_I will define the state set and transitions of M^C.

As M is r-reduced there is some characterizing set W that distinguishes the states of M. This need not, however, distinguish the states of M_I. The following section will consider the problem of generating some set Z with the property that: if I conforms to M and $M_I \in \Phi_M^m$ then $W \cup Z$ is a characterizing set for M_I. Z will be used when generating M^C. Section 4.2 then gives the algorithm while Section 4.3 contains a proof of correctness. Section 4.4 discusses possible optimizations and this is followed by an example. Before exploring the problem of generating the candidate, two useful results will be given. The first of these can be found in [15].

Lemma 1 *Suppose $M = (S, s_1, h, \Sigma, \Gamma)$ is an r-reduced ONFSM and $M' = (A, a_1, \delta, \lambda, \Sigma, \Gamma)$ is an initially connected DFSM that conforms to M. Then for every state $a \in A$, there is some state $s \in S$ such that a conforms to s.*

The following properties, of any DFSM that conforms to M, will be used later.

Lemma 2 *Suppose $M = (S, s_1, h, \Sigma, \Gamma)$ is an r-reduced ONFSM and $M' = (A, a_1, \delta, \lambda, \Sigma, \Gamma)$ is an initially connected DFSM that conforms to M. Then for every input sequence $x \in \Gamma^*$, the state $a = \delta^*(a_1, x)$ conforms to the state s with $\{s\} = h_{\lambda^*(a_1,x)}(s_1, x)$.*

Proof

Proof by contradiction: suppose this does not hold for some $x \in \Gamma^*$. Since M' conforms to M, by Lemma 1 $a = \delta^*(a_1, x)$ must conform to some state $s' \in S \setminus \{s\}$.

Since M is r-reduced and a conforms to s', there is some input sequence w such that $\lambda^*(a, w) \notin h_2^*(s, w)$. Now consider $\lambda^*(a, xw)$. Since M is observable, every element of $h_2^*(s_1, xw)$ that starts with $\lambda^*(a_1, x)$ must be an element of $\lambda^*(a_1, x)h_2^*(s, w)$. But, $\lambda^*(a, w) \notin h_2^*(s, w)$ and thus $\lambda^*(a_1, xw) \notin h_2^*(s_1, xw)$. This contradicts M' conforming to M, as required. □

4.1 Distinguishing the states

In order to produce the state set of M^C it is necessary to be able to distinguish states. However, recall that the candidate M^C need not represent I: it must have the property that if $M_I \in \Phi_M^m$ then I is correct if and only if M_I is equivalent to M^C. It will prove sufficient to base the search on tests that are guaranteed to distinguish between any two non-equivalent states of any DFSM from Φ_M^m that conforms to M. This section will consider the problem of producing such tests.

Recall that T is the state set of the unknown DFSM M_I. Given a state s of M, let $Q(s)$ denote the set of states of M_I that conform to s: $Q(s) = \{t \in T | t \leq s\}$. As M is r-reduced, if M_I conforms to M, then the set of $Q(s)$ partitions T.

Since M is r-reduced, W distinguishes between any two states t and t' of M_I with $t \in Q(s)$ and $t' \in Q(s')$ for some $s' \neq s$. It is sufficient to develop a set Z of input sequences that distinguishes the elements within each $Q(s)$. As M_I is reduced there must be such a set.

Inspired by [5] two states t, t' of M_I will be said to be *p_M-equivalent* if there is some $s \in S$ such that $t, t' \in Q(s)$ and no input sequence of length p or less distinguishes t and t'; otherwise they are *p_M-separable*. This defines an equivalence relation $\approx_p$: $t \approx_p t'$ if and only if t and t' are p_M-equivalent. Note that this definition crucially relies on the $Q(s)$ partitioning the state set of any DFSM that conforms to M. In turn, this property is a consequence of M being r-reduced.

Let k denote an upper bound on the number of states of M_I not reached by V_{det} if M_I conforms to M. Let $n' = |V_{det}|$. Thus, $k = m - n'$ and if M is d-connected then $k = m - n$. The following results demonstrate that for all $s \in S$, $t, t' \in Q(s)$, $t \neq t'$, t and t' are k-separable and thus that $Z = \Sigma^k$ suffices. In Section 4.4 the problem of reducing this, by using some set of initial sequences from Σ^k, will be considered.

Lemma 3 *Let M denote an r-reduced ONFSM and let M_I be a reduced DFSM that conforms to M. Suppose M_I contains two p_M-equivalent states t_i and t_j. Then there are states t'_i, t'_j of M_I that are p_M-equivalent but $(p+1)_M$-separable.*

Proof

Let h denote the state transition function of M and δ_I and λ_I represent the next state and output functions, respectively, of M_I.

Let x denote a shortest input sequence that distinguishes t_i and t_j. Then $|x| > p$. Let x' denote the first $|x| - p - 1$ elements of x and thus $x = x'x''$ for some x''.

As t_i and t_j are p_M-equivalent, there is some state s of M such that t_i and t_j conform to s.

Consider the states $t'_i = \delta_I^*(t_i, x')$ and $t'_j = \delta_I^*(t_j, x')$. Let $y = \lambda_I^*(t_i, x') = \lambda_I^*(t_j, x')$. As M is observable the final state of M, after x' has been input in state s leading to output y, is unique. Let this state be denoted s': $\{s'\} = h_y^*(s, x')$.

As $t_i, t_j \leq s$, $\forall \alpha \in \Sigma^*.\lambda_I^*(t_i, x'\alpha), \lambda_I^*(t_j, x'\alpha) \in h_2^*(s, x'\alpha)$.

Therefore, $\forall \alpha \in \Sigma^*.y\lambda_I^*(t'_i, \alpha), y\lambda_I^*(t'_j, \alpha) \in \{y\}h_2^*(s', \alpha)$.

Thus, $\forall \alpha \in \Sigma^*.\lambda_I^*(t'_i, \alpha), \lambda_I^*(t'_j, \alpha) \in h_2^*(s', \alpha)$ and thus $t'_i, t'_j \leq s'$.

By the minimality of x, t'_i and t'_j are distinguished by x'' but by no shorter input sequence. But $|x''| = p+1$ and thus t'_i and t'_j are $(p+1)_M$-separable. As t'_i and t'_j are distinguished by no shorter sequence and t'_i, $t'_j \leq s'$, t'_i and t'_j are also p_M-equivalent, as required. □

Theorem 4 *Let M denote an r-reduced ONFSM with n states and let M_I be a reduced DFSM with at most m states. Suppose V_{det} is a deterministic state cover that reaches n' states of M. Further, suppose that M_I conforms to M and let $k = m - n'$. Given states $t \neq t'$ of M_I, such that $t, t' \in Q(s)$ for some $s \in S$, there is some input sequence x, of length at most k, that distinguishes t and t'.*

Proof

Suppose x is a shortest input sequence that distinguishes t and t'. Then t and t' are 0_M-equivalent, 1_M-equivalent, …, $(|x|-1)_M$-equivalent. By Lemma 3 the equivalence relations $\approx_0$, $\approx_1$, $\approx_2$, …, $\approx_{|x|}$ are distinct. In general, for $0 \leq p < |x|$, $\approx_{p+1}$ is a proper refinement of $\approx_p$ and thus $\approx_{p+1}$ has more equivalence classes than $\approx_p$. It is now sufficient to note that $\approx_0$ has at least n' equivalence classes (as $\sim_W$ has at least n' equivalence classes) and $\approx_{|x|}$ has at most m equivalence classes. There can thus have been at most $m - n'$ (proper) refinements and thus $|x| \leq k$ as required. □

Thus, for all $s \in S$ the set $Z = \Sigma^k$ is guaranteed to distinguish the elements of $Q(s)$.

1. Test I with Z and let Y denote the corresponding set of input/output sequences. If the behaviour observed is not consistent with M then terminate.

2. Set $V = \{\epsilon\}$, $U = \{u_1\}$, $v_{u_1} = \epsilon$, $o(u_1) = Y$, $j = 1$, $Q(s_1) = \{u_1\}$, $\forall s \in S \setminus \{s_1\}.Q(s) = \emptyset$, $E_1 = \{\epsilon\}$, and $E_2 = \{\epsilon\}$.

3. If for all $x \in \Sigma$, $v \in V$, vx is in E_1, then terminate else choose some $v' \in V\Sigma \setminus E_1$.

4. Test I with $\{v'\}Z$. If the behaviour observed is not consistent with M then terminate.

5. Let $v' = v_u x$ for some $x \in \Sigma$ and $u \in U$. Suppose the response of I to v' is $y = y_1 y_2$, $y_2 \in \Gamma$. Add v'/y to E_2 and add v' to E_1. Let Y denote the input/output behaviour observed by the use of Z after v' and suppose $\{s'\} = h^*_y(s_1, v')$. If there exists some $u_p \in Q(s')$ such that $Y = o(u_p)$ then

   set $\delta_C(u, x) = u_p$, set $\lambda_C(u, x) = y_2$, and goto 3

   else

   add u_{j+1} to both U and $Q(s')$, add v' to V, set $v_{u_{j+1}} = v'$, set $o(u_{j+1}) = Y$, set $\delta_C(u, x) = u_{j+1}$, set $\lambda_C(u, x) = y_2$, increase j by 1, and goto 3

Figure 3: Algorithm 1

4.2 Generating the candidate

This section will give an algorithm for generating the candidate M^C. The algorithm applies a breadth first search, distinguishing states of M_I. Since M is r-reduced and observable, if M_I conforms to M then a state of M_I cannot conform to two different states of M. Further, since M is observable, an input/output sequence observed in testing determines the corresponding state of M reached. It will thus transpire that, in the algorithm, it is sufficient to use Z to distinguish states of M_I.

This algorithm, which is given in Figure 3, will now be described informally. In this algorithm V denotes the current state cover and U denotes the corresponding set of states of M^C. Given state $u \in U$, v_u denotes the sequence from V that reaches u and $o(u)$ denotes the output produced when Z is input in the state of M_I reached by v_u. For each state $s \in S$, $Q(s)$ denotes the set of states from U that correspond to s. E_1 denotes the set of input sequences whose final state has been explored in the search and E_2 denotes the corresponding set of input/output sequences. j denotes the number of states found to date ($j = |U|$).

Z is used to generate a state cover V^C for M^C based on a breadth-first search of the states of M_I. This search terminates once each state of M_I, reached by extending a sequence from $V = V^C$ by one input value, is equivalent, under Z, to some state from the same $Q(s)$ that has already been met. This also defines the transitions of M^C: the response to xz, for each $z \in Z$, after $v \in V^C$ determines the corresponding transition of M^C.

Note that while $Z \cup W$ is a characterising set for M^C, only Z need be used in the search. This is because, since M is r-reduced, the state set of M^C is partitioned into the $Q(s)$: no state of M^C may conform to two different state of M. Given a state met by input/output sequence x/y in testing, this state is placed in the $Q(s)$ that has $\{s\} = h_y(s_1, x)$. Thus, it is sufficient to distinguish between the elements of each $Q(s)$ and Z is guaranteed to do this.

Possible optimisations to the algorithm are discussed in Section 4.4. These include the addition of an alternative termination criterion: the algorithm may terminate if m separate states have been found. However, a little more effort may then be required: it may be necessary to determine some of the transitions leaving states reached on the final iteration.

The next section contains a proof of correctness for Algorithm 1. This is followed by some possible optimizations and then an example.

4.3 Proof of correctness

The following two Lemmas will be used in the proof of the correctness of Algorithm 1. The first shows that if there is a mapping, often called a *simulation relation* [12], from the states of a DFSM M' to the states of an

NDFSM M, then M' conforms to M.

Lemma 5 *Suppose $M = (S, s_1, h, \Sigma, \Gamma)$ is a completely specified ONFSM, $M' = (A, a_1, \delta, \lambda, \Sigma, \Gamma)$ is a completely specified DFSM and there exists a function $f : A \rightarrow S$ with $f(a_1) = s_1$ such that for all $a_i, a_j \in A$, $x \in \Sigma$, and $y \in \Gamma$, if $(a_i, a_j, x/y)$ is a transition of M' then $(f(a_i), f(a_j), x/y)$ is a transition of M. Then M' conforms to M.*

Proof

Proof by contradiction: suppose M' does not conform to M. Thus there exists some $x \in \Sigma^*$ such that $\lambda^*(a_1, x) \notin h_2^*(s_1, x)$. Consider some minimal length input sequence x such that

$$(f(\delta^*(a_1, x)), \lambda^*(a_1, x)) \notin h^*(s_1, x)$$

Clearly $|x| > 0$. Let $x = x_1 x_2$ for some $x_2 \in \Sigma$ and let $a = \delta^*(a_1, x_1)$, $a' = \delta(a, x_2)$, $s = f(a)$, $s' = f(a')$, $y_1 = \lambda^*(a_1, x_1)$, and $y_2 = \lambda(a, x_2)$.

Now consider the transition $(a, a', x_2/y_2)$ of M'. Then $(f(a), f(a'), x_2/y_2)$ is a transition of M. Also, by the minimality of x, $(f(a), y_1) = (f(\delta^*(a_1, x_1)), \lambda^*(a_1, x_1)) \in h^*(s_1, x_1)$. Thus $(f(a'), y_1 y_2) \in h^*(s_1, x_1 x_2)$ and so $(f(\delta^*(a_1, x)), \lambda^*(a_1, x)) \in h^*(s_1, x)$. This provides a contradiction as required. □

Lemma 6 *Suppose candidate M^C is produced by the application of Algorithm 1 and that the process of generating M^C identifies no failures. Then there exists a function $f : U \rightarrow S$ with $f(u_1) = s_1$ such that if $(u, u', x/y)$ is a transition of M^C then $(f(u), f(u'), x/y)$ is a transition of M.*

Proof

The function f will be defined on the basis of the search. Given $u \in U$, if $u \in Q(s)$ then $f(u) = s$. Clearly $f(u_1) = s_1$.

Consider a transition $(u, u', x/y)$ of M^C with $f(u) = s$ and $f(u') = s'$. Observe that the transition from u with input x is derived in Algorithm 1 by following $v_u x$ by Z. Further, the state u' must satisfy $u' \in Q(s')$ for $s' = h_y(s, x)$. Thus, $(f(u), f(u'), x/y)$ is a transition of M as required. □

Theorem 7 *Suppose M is an r-reduced ONFSM and the IUT I satisfies the test hypotheses. Further, suppose Algorithm 1 generates the candidate M^C and no failures are found in this process. Then I conforms to M if and only if M_I is equivalent to M^C.*

Proof

Case 1: $\Leftarrow$

This follows immediately from Lemmas 5 and 6.

Case 2: $\Rightarrow$

Proof by contradiction: suppose I conforms to M but M_I is not equivalent to M^C. Note that by Lemmas 5 and 6, M^C conforms to M. Define a function $g : U \rightarrow T$ by: given $u \in U, v \in V^C$ with $\delta_C^*(u_1, v) = u$, $g(u) = \delta_I^*(t_1, v)$.

Let x be an input sequence, that is a minimal length extension of an element of V^C, such that one of the following hold:

1. $\lambda_I^*(t_1, x) \neq \lambda_C^*(u_1, x)$
2. $\delta_I^*(t_1, x) \neq g(\delta_C^*(u_1, x))$

Since M_I and M^C are not equivalent there must be some such x. By construction, $x \notin V^C$. Then $x = x_1 x_2$ for some $x_2 \in \Sigma$ with x_1 being some extension to a sequence in V^C. Let v denote the element of V^C such that $\delta_C^*(u_1, v) = \delta_C^*(u_1, x_1)$.

By the minimality of x, $g(\delta_C^*(u_1, v)) = \delta_I^*(t_1, v)$ and $g(\delta_C^*(u_1, x_1)) = \delta_I^*(t_1, x_1)$. Since $\delta_C^*(u_1, v) = \delta_C^*(u_1, x_1)$, $\delta_I^*(t_1, v) = \delta_I^*(t_1, x_1)$. Let t denote $\delta_I^*(t_1, v)$ and u denote $\delta_C^*(u_1, v)$. Observe that either $\lambda_I(t, x_2) \neq \lambda_C(u, x_2)$ or $\delta_I(t, x_2) \neq g(\delta_C(u, x_2))$.

First suppose that $\lambda_I(t, x_2) \neq \lambda_C(u, x_2)$. This contradicts the fact that the output of the transition from u with input x_2 is based on the response of I to vx_2. Thus $\lambda_I(t, x_2) = \lambda_C(u, x_2)$ and so $\delta_I(t, x_2) \neq g(\delta_C(u, x_2))$. Let $y_1 = \lambda_I^*(t_1, v)$ and $y_2 = \lambda_I(t, x_2)$. Let s, s' be the unique states of M such that $\{s\} = h_{y_1}^*(s_1, v)$ and

$\{s'\} = h_{y_2}(s, x_2)$. Since M_I and M^C produce the same output on vx_2 and M^C and M_I conform to M, by Lemma 2, $t, u \leq s$ and $\delta_I(t, x_2), \delta_C(u, x_2) \leq s'$.

Since $\delta_I(t, x_2) \neq g(\delta_C(u, x_2))$, the state $u' = \delta_C(u, x_2)$ is reached by some sequence $v_{u'} \in V^C$ with $v_{u'} \neq vx_2$. By construction, M_I and M^C produce the same response to input $v_{u'}$. Thus, since $u' \leq s'$, $\delta_I(t_1, v_{u'}) \leq s'$.

Since $\delta_C(u, x_2) = u'$, the response of I to Z after $v_{u'}$ and vx_2 is identical. Since I conforms to M, Z pairwise distinguishes the states of M_I that conform to s' and so $\delta_I^*(t_1, v_{u'}) = \delta_I^*(t_1, vx_2) = \delta_I(t, x_2)$. But $\delta_C^*(u_1, v_{u'}) = \delta_C^*(u_1, vx_2) = \delta_C(u, x_2)$ and, by the definition of g, $g(\delta_C^*(u_1, v_{u'})) = \delta_I^*(t_1, v_{u'})$. Thus $\delta_I(t, x_2) = g(\delta_C(u, x_2))$, providing a contradiction as required. □

4.4 Optimizations

This section will briefly consider approaches that might reduce the test effort required to generate M^C. First, it is possible to add a new termination criterion: when m states have been found. If this condition is satisfied, testing should terminates immediately if all transitions have been determined. Otherwise a final phase of testing determines any remaining transitions. The test effort might also be reduced by starting with $V = V_{det}$.

It is possible to reduce the set Z by using a separate set Z_i for each $s_i \in S$. The following proposition gives a further condition that may be utilised in order to reduce the test effort.

Proposition 8 *Given state $s_i \in S$, it is only necessary to include, within Z_i, input sequences that can lead to a sequence of transitions from M that end in a non-deterministic transition.*

This condition comes from the fact that if an input/output sequence x/y is executed from some state in $Q(s_i)$ then deterministic transitions at the end of the corresponding sequence from M don't help distinguish this state from any other element of $Q(s_i)$.

4.5 Example

This section will consider testing from M_0 when I behaves like some DFSM M_I that may have up to $n + 2$ states (in this case $n = 3$). Thus, as M is d-connected, $k = 2$ and thus Σ^2 is guaranteed to distinguish the states within each $Q(s_i)$. This section will utilize some of the optimizations discussed earlier.

The set of sequences that leave s_1, have length at most 2 and end in a non-deterministic transition is contained in:

$\{a/1b/0,\ a/1b/1,\ a/0\}$

Thus $Z_1 = \{ab\}$. If the response to a is 0, this need not be followed by b.

The corresponding set of sequences for s_2 is:

$\{b/1a/0,\ b/1a/1,\ a/0b/0,\ a/0b/1\}$

Thus $Z_2 = \{ba,\ ab\}$.

Finally, for s_3, the set is:

$\{b/0a/0,\ b/0a/1,\ b/1,\ a/1b/0,\ a/1b/1\ \}$

Thus $Z_3 = \{ba,\ ab\}$. If the response to b is 1 then it need not be followed by a.

Suppose the response of I to ab is 10. This identifies $u_1 = u_1^1$ within $Q(s_1)$. Here the notation will be: u_i denotes the ith state found in the search while u_i^j denotes the jth element of $Q(s_i)$ found. Now consider the state after $a/1$. Z_3 will be used to identify the state reached by $a/1$. Suppose the response of I, after input a, to ba is 01 and the response after a, to ab, is 10. Let $u_2 = u_3^1 = \delta_C(u_1, a)$. Thus $\lambda_C^*(u_3^1, ba) = 01$.

Let $u_3 = u_2^1$ denote $\delta_C(u_1^1, b)$ and suppose that the response of I to bba is 011 and the response of I to bab is 001. This is sufficient to identify u_2^1 within $Q(s_2)$.

Now consider the state of M^C after the execution of ba: this is an element $u_4 = u_3^2 = \delta_C^*(u_1, ba)$ of $Q(s_3)$. Suppose the response of I to bab is 001: by the rule for Z_3, the value a need not be input after this. Further, suppose that the response to $baab$ is 0011. Clearly $u_3^2 \neq u_3^1$. At this stage the process has derived the section of M^C and information shown in Figure 4.

Consider now the response of I to bb: the state reached is in $Q(s_1)$. Suppose this state also responds to ab with 10. Thus, from u_2^1 the input of b moves M^C to state u_1^1.

There are two more input sequences of length 2 to consider: aa and ab. Suppose ab is considered: let $u_5 = u_1^2 = \delta_C^*(u_1, ab) \in Q(s_1)$. Suppose the response of I to $abab$ is 1011. Thus, $\lambda_C^*(u_1^2, ab) = 11 \neq \lambda_C^*(u_1^1, ab)$.

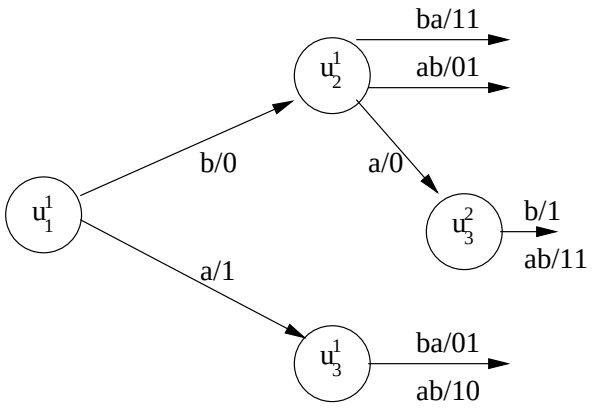

Figure 4: Part of M^C

Thus $u_1^2 \neq u_1^1$. The search has now found 5 different states: two elements from each of $Q(s_1)$ and $Q(s_3)$ and one element from $Q(s_2)$. Thus, as $m = 5$, a state cover has been found for M^C. This is:

$$\{\epsilon, b, a, ba, ab\}.$$

The following input sequences distinguish the elements of the $Q(s_i)$:

- States in $Q(s_1)$: ab and responses 10 (u_1^1) or 11 (u_1^2).
- States in $Q(s_2)$: no input required as $|Q(s_2)| = 1$.
- States in $Q(s_3)$: b and responses 0 (u_3^1) and 1 (u_3^2).

Consider, now, the transition defined by initial state u_3^1 and input a. The final state is an element of $Q(s_3)$ and thus must be followed by b. Suppose output 0 is generated when b is input in state $\delta_C(u_3, a)$. Thus, the final state of this transition is u_3^1.

Suppose the following behaviour is exhibited:

- From state u_1^2: $a/1b/1$ (so $\delta_C(u_1^2, a) = u_3^2$)
- From state u_3^2: $a/1b/1$ (so $\delta_C(u_3^2, a) = u_3^2$)
- From state u_3^1: $a/1b/0$ (so $\delta_C(u_3^1, a) = u_3^1$)

All other transitions either have already been checked or are deterministic and end at an element of $Q(s_2)$ (and so don't need checking, as $|Q(s_2)| = 1$).

This resolves the non-determinism and, under the test hypotheses, I conforms to M if and only if M_I is equivalent to the DFSM given in Figure 5.

5 Testing when M is not r-reduced

This section will extend the approach, given in Section 4, to the general case where M is not r-reduced. The approach will be similar to that given for when M is r-reduced: a breadth-first search through the states of M_I will be performed.

The case considered in this section is significantly different from the case where M is r-reduced. This is because a state t of a DFSM M_I that conforms to M may conform to more than one state of M. Thus, even if I conforms to M the $Q(s)$ need not partition the states of M_I.

The following section will consider the problem of distinguishing the states of M_I. This will be followed by an algorithm for generating a candidate. Section 5.3 contains a proof of correctness. Section 5.4 will then describe some possible optimizations.

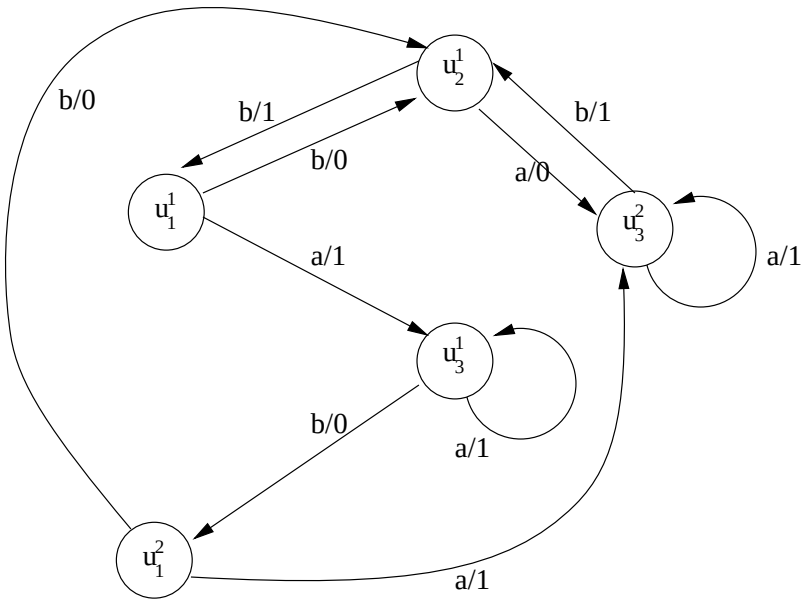

Figure 5: The Candidate

5.1 Distinguishing States

Two states t, t' of M_I will be said to be (p, W)-*equivalent* if for all $w \in W$ and $x \in \Sigma^*$ such that $0 \leq |x| \leq p$, $\lambda^*(t, xw) = \lambda^*(t', xw)$, and otherwise they are (p, W)-*separable*. $t \approx_{p,W} t'$ will denote t and t' being (p, W)-equivalent and otherwise $t \not\approx_{p,W} t'$. Clearly $\approx_{p,W}$ is an equivalence relation on the states of M_I. An alternative formalisation of this definition is that t and t' are (p, W)-equivalent if and only if for all $x' \in (\{\epsilon\} \cup \Sigma \cup \ldots \cup \Sigma^p)W$, $\lambda^*(t, x') = \lambda^*(t', x')$.

The set $(\Sigma \cup \{\epsilon\})^k = (\{\epsilon\} \cup \Sigma \cup \ldots \cup \Sigma^k)$ denotes the set of sequences, of elements from Σ, of length at most k. Suppose W and V are finite sets of input sequences such that V reaches n' states of M_I that are pairwise distinguished by W (V need not be V_{det} and W need not be a characterizing set for M). Let $k = m - n'$. With the above definition of k, it is now possible to prove that the set $(\Sigma \cup \{\epsilon\})^k W = (\{\epsilon\} \cup \Sigma \cup \ldots \cup \Sigma^k)W$ is guaranteed to distinguish any two non-equivalent states of M_I if I conforms to M on VW.

Lemma 9 *Let $W \subset \Sigma^*$ be a set of input sequences and $M_I = (T, t_1, \lambda_I, \delta_I, \Sigma, \Gamma)$ be a reduced DFSM. Suppose M_I contains two states t_i, t_j that are (p, W)-equivalent. Then there are states t_i' and t_j' of M_I that are (p, W)-equivalent but $(p+1, W)$-separable.*

Proof

Let x denote a shortest input sequence that has some $w \in W$ such that xw distinguishes t_i and t_j. Further, let w denote some element of W such that xw distinguishes t_i and t_j.

Then, as t_i and t_j are (p, W)-equivalent, $|x| > p$. Let x'' denote the last $p+1$ elements of x and thus $x = x'x''$ for some x'.

By the minimality of x, $\lambda^*(t_i, x') = \lambda^*(t_j, x')$. Consider the states $t_i' = \delta^*(t_i, x')$ and $t_j' = \delta^*(t_j, x')$. Then, as t_i and t_j are distinguished by $x'x''w$ and $\lambda^*(t_i, x') = \lambda^*(t_j, x')$, t_i' and t_j' are distinguished by $x''w$. Thus, t_i' and t_j' are $(p+1, W)$-separable.

By the minimality of x, for all z with $|z| < |x''|$ and $w' \in W$, $\lambda^*(t_i', zw') = \lambda^*(t_j', zw')$. Thus, as $|x''| = p+1$, t_i' and t_j' are (p, W)-equivalent.

To conclude, t_i' and t_j' are (p, W)-equivalent and $(p+1, W)$-separable as required. □

The proof of the following result is equivalent to that of Theorem 4.

Theorem 10 *Let M_I denote a reduced DFSM with input alphabet Σ and at most m states. Let $W \subset \Sigma^*$ and $V \subset \Sigma^*$ be any pair of sets of input sequences. Suppose that the states of M_I reached by V are pairwise*

1. Set $V_1 = \{\epsilon\}$, $U_1 = \{u_1\}$, $W_1 = W$, $k_1 = m - 1$, $v_{u_1} = \epsilon$, $i = 1$, and $j = 1$.

2. Set $S' = \widehat{S}_1$, where S_1 is some set of r-distinguishable states of M. Repeat until $S' = \emptyset$, choose $s \in S'$ and some $v \in V_{det}$ such that $h_1^*(s_1, v) = \{s\}$. Set $S' = S' \setminus \{s\}$, $U_1 = U_1 \cup \{u_{j+1}\}$, $v_{u_{j+1}} = v$, $V_1 = V_1 \cup \{v\}$, $k_1 = k_1 - 1$, and $j = j + 1$.

3. If $i = 1$ test with $V_1(\Sigma \cup \{\epsilon\})^{k_1+1}W_1$ else test with $V_i(\Sigma \cup \{\epsilon\})^{k_i+1}W_i \setminus V_{i-1}(\Sigma \cup \{\epsilon\})^{k_{i-1}+1}W_{i-1}$. If the behaviour observed in testing is not consistent with M then terminate. For all $v \in V_i$ and $x \in \Sigma$ let $o(v, i)$ denote the response of I to $(\Sigma \cup \{\epsilon\})^{k_i}W_i$ after v and $o(vx, i)$ denote the response of I to $(\Sigma \cup \{\epsilon\})^{k_i}W_i$ after vx. Let $V_{i+1} = V_i$, $U_{i+1} = U_i$, and $W_{i+1} = W_i$.

4. For all $v_u \in V$, $x \in \Sigma$, where I responds to $v_u x$ with $y_1 y_2$ ($y_2 \in \Gamma$) do the following:

   If $o(vx, i) = o(v_{u'}, i)$ for some $v_{u'} \in V_{i+1}$ then $\delta_C(u, x) = u'$ and $\lambda_C(u, x) = y_2$

   else

   $U_{i+1} = U_{i+1} \cup \{u_{j+1}\}$, $v_{u_{j+1}} = v_u x$, $V_{i+1} = V_{i+1} \cup \{v_u x\}$, and $j = j + 1$. Choose some $X \subseteq (\Sigma \cup \{\epsilon\})^{k_i}W_i$ such that $W_{i+1} \cup X$ distinguishes the state of M_I reached by $v_u x$ from all other states of M_I reached by sequences in $V_{i+1} \setminus \{v_u x\}$. Set $W_{i+1} = W_{i+1} \cup X$, $\delta_C(u, x) = u_{j+1}$, and $\lambda_C(u, x) = y_2$.

5. Set $i = i + 1$ and $k_i = m - |V_i|$.

6. If $V_i = V_{i-1}$ or $|V_{i-1}| \geq m$ then terminate else goto step 3.

Figure 6: Algorithm 2

distinguished by W and $|V| = n'$. Let $k = m - n'$. Given states t and t' of M_I, $t \neq t'$, there is some input sequence x, of length at most k, and $w \in W$ such that xw distinguishes t and t'.

5.2 Finding the Candidate

It is now possible to generate the candidate, M^C, using a breadth first search. The algorithm, for generating M^C, is given in Figure 6. This will now be informally described.

At each step the test $V_i(\Sigma \cup \{\epsilon\})^{k_i+1}W_i$ is used. This will identify the states reached by transitions that leave states in the image of V_i (in M_I). If m states have been found or no state of M_I reached by $V_i\Sigma$ is distinguished, by $(\Sigma \cup \{\epsilon\})^{k_i}W_i$, from the states of M_I reached by V_i then the search terminates. If new states are reached, for each of these an input sequence from $V_i\Sigma$, that reaches this state, is identified. V_{i+1} is formed by adding the set of such sequences to V_i, $k_{i+1} = m - |V_{i+1}|$. Finally, if necessary W_i is updated to form W_{i+1}, so that it distinguishes the states of M_I that are reached by V_{i+1}.

Step 1 initializes some variables and Step 2 essentially builds a set of states corresponding to a pairwise distinguishable subset of the d-reachable states of M (recall that $\hat{D} = D \cap S_{V_{det}}$ where $S_{V_{det}}$ is the set of states of states of M reached by sequences from V_{det}). The iterative process then begins. The algorithm terminates when either no new states are met ($V_i = V_{i-1}$) or when m state were found in the previous iteration ($|V_{i-1}| \geq m$). The algorithm does not terminate immediately once m states are found since the transitions from the states met on the most recent iteration need not have been determined. Note that once m states have been found, the final iteration could be directed at the transitions yet to be determined. This approach further reduces the test effort but has not been included in the algorithm in order to aid clarity. The final values V_r, W_r and k_r will be denoted V^C, W^C, and k_c respectively.

At each step either the process terminates or the value of k is reduced by *at least* one and the length of sequences in V is increased by at most 1. Suppose the process terminates after r steps. For all i, $0 \leq i < r$, $V_{i+1}(\Sigma \cup \{\epsilon\})^{k_{i+1}} \subseteq V_i(\Sigma \cup \{\epsilon\})^{k_i}$. Thus each of the above steps uses a test contained in $V_{det}(\Sigma \cup \{\epsilon\})^{m-n'+1}W^C$ where $n' = |\hat{S}_1|$.

Not only does this search find a state set of M^C, it also determines the transition set. Thus, the algorithm defines a DFSM M^C. The test used to generate the candidate contains that used when applying the W-method on the basis of the candidate. This test is thus complete: a second phase of testing is not required.

It will be proved, in the next section, that if I satisfies the test hypotheses then I behaves like M^C. There thus

remains one step: to decide whether M^C conforms to M. This involves determining whether $L(M^C) \subseteq L(M)$ which may be achieved in polynomial time by producing a FA that recognises the language $L(M^C) \setminus L(M)$ (see, for example, [3]) and then deciding whether this language is empty.

The following section will prove that, under the test hypotheses, I is correct if and only if M^C conforms to M and I behaves like M^C.

5.3 Proof of correctness

The following is the key result.

Theorem 11 *Suppose M is a reduced ONFSM and the IUT I satisfies the test hypotheses. Further, suppose Algorithm 2 generates the candidate M^C and no failures are found in this process. Then I behaves like M^C.*

Proof

Proof by contradiction: suppose I behaves like some DFSM $M_I \in \Phi_M^m$ that is not equivalent to M^C. Define a function $g : U \to T$ by: $g(u) = \delta_I^*(t_1, v_u)$ and suppose x is a minimal length extension of some sequence in V^C, such that one of the following holds:

1. $\lambda_C^*(u_1, x) \neq \lambda_I^*(t_1, x)$
2. $g(\delta_C^*(u_1, x)) \neq \delta_I^*(t_1, x)$

Since M_I is not equivalent to M^C, there is some such x. Then $x = x_1x_2$ for some $x_2 \in \Sigma$. Clearly $x \notin V^C$.

Let $u = \delta_C^*(u_1, x_1)$, $u' = \delta_C(u, x_2)$, $t = \delta_I^*(t_1, x_1)$, and $t' = \delta_I(t, x_2)$. By the minimality of x, $g(\delta_C^*(u_1, x_1)) = \delta_I^*(t_1, x_1)$ and $g(\delta_C^*(u_1, v_u)) = \delta_I^*(t_1, v_u)$. Thus, since $\delta_C^*(u_1, x_1) = \delta_C^*(u_1, v_u)$, $t = \delta_I^*(t_1, x_1) = \delta_I^*(t_1, v_u)$.

Observe that one of the following must hold:

1. $\lambda_C(u, x_2) \neq \lambda_I(t, x_2)$
2. $g(\delta_C(u, x_2)) \neq \delta_I(t, x_2)$

Suppose that in the generation of M^C, v_u first appears, as part of the state cover, in V_{i_1}. Then v_u and v_ux_2 are both used in testing and followed by $Z^{i_1} = (\Sigma \cup \{\epsilon\})^{k_{i_1}} W_{i_1}$. Observe that, by construction, $\lambda_C(u, x_2) = \lambda_I(\delta_I^*(t_1, v_u), x_2) = \lambda_I(t, x_2)$. This contradicts the first of the above possibilities.

Suppose that, when v_ux_2 is used in testing, to determine $\delta_C(u, x_2)$, the state u' is not already in U. Thus $v_{u'} = v_ux_2$. Thus, by the definition of g, $g(\delta_C(u, x_2)) = \delta_I(t, x_2)$. This contradicts the second condition. Thus, when v_ux_2 is used in testing, to determine $\delta_C(u, x_2)$, the state u' is already in U. Since, by Theorem 10, Z^{i_1} pairwise distinguishes the states of M_I, $\delta_I^*(t_1, v_ux_2) = \delta_I^*(t_1, v_{u'})$. But $\delta_I^*(t_1, v_ux_2) = \delta_I(t, x_2)$ and thus $\delta_I(t, x_2) = \delta_I^*(t_1, v_{u'}) = g(\delta_C^*(u_1, v_{u'})) = g(\delta_C^*(u_1, v_ux_2)) = g(\delta_C(u, x_2))$. This provides a contradiction as required.

□

Theorem 12 *Suppose M is a reduced ONFSM and the IUT I satisfies the test hypotheses. Further, suppose Algorithm 2 generates the candidate M^C, no failures are found in this process, and M^C conforms to M. Then I conforms to M if and only if I behaves like M^C.*

Proof

This follows immediately from Theorem 11. □

5.4 Optimizations

Once a value of k has been found it is possible to extend the notion of r-distinguishing states to deterministically (V, k) distinguishing states [8]. The use of deterministically (V, k) distinguishing as opposed to r-distinguishing may, naturally, increase the size of the maximal set of pairwise distinguishable states. This in turn, reduces k and thus alters the form of distinguishability used. This process may go through at most $|S| - |\widehat{S_1}|$ iterations, leading to a final value of k. A characterizing set W is associated with this value of k.

Interestingly, Algorithm 2 achieves more than is required: it generates a model M^C of I based only on the assumption that $M_I \in \Phi_M^m$. This suggests that there may be alternative algorithms, that require less testing,

that produce a model M^C such that, under the hypotheses, I conforms to M if and only if I behaves like M^C. Such algorithms will be a topic of future work.

Finally, as noted, where the algorithm terminates due to m states being found the final iteration may be reduced. This is because this final iteration simply determines the transitions from each state met for the first time in the previous iteration.

6 Comparison with a previous method

The approach, of generating a candidate when testing a deterministic implementation against an NFSM, is new. Once M^C has been produced, any algorithm for generating tests from a deterministic FSM may be applied. However, it is possible to consider the test T_1 produced, when M^C is generated and then the W-method is applied. This section will compare the size of T_1 with that of the test T_2 produced using the approach of [15] when the entire characterizing set is used for state identification. However, it is worth noting that these approaches have slightly different objectives.

Suppose M has n states, M_I has at most m states and M^C has m_c states. As before, let V_{det} and W denote a deterministic state cover and a characterizing set for M and V^C and W^C denote extensions of V_{det} and W that provide a state cover and characterizing set for M^C. Let $D_1, \ldots, D_p$ denote maximal sets of pairwise r-distinguishable states from M and let d denote the size of the largest $\widehat{D}_i$. Let $k = m - d$ and $k_c = m - m_c$.

In the generation of M^C, the test used is contained in $V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W^C$. In the application of the W-method the test produced is $V^C(\Sigma \cup \{\epsilon\})^{k_c+1}W^C$. It is known that $|V^C| - |V_{det}| \leq m_c - d = k - k_c$ and each sequence in V^C is at worst a sequence from V_{det} extended by a sequence of length at most $|V^C| - |V_{det}|$. Thus $V^C(\Sigma \cup \{\epsilon\})^{k_c+1} \subseteq V_{det}(\Sigma \cup \{\epsilon\})^{|V^C|-|V_{det}|}(\Sigma \cup \{\epsilon\})^{k_c+1} \subseteq V_{det}(\Sigma \cup \{\epsilon\})^{(k-k_c)+k_c+1} = V_{det}(\Sigma \cup \{\epsilon\})^{k+1}$ and so, assuming the W-method is used, T_1 is contained in $V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W^C$.

Consider now the algorithm given in [15]. Here, an input sequence x is maximal (in a traversal set for state s) if *all* possible sequences of transitions from state s of M, using x, meet some D_i at least $m - |\widehat{D}_i| + 1$ times. Thus, the sequences have length at least $m - d + 1 = k + 1$. Naturally, where d is smaller than n, many of the sequences will be significantly longer than this. This is because, in order to meet states from D_i at least $m - |\widehat{D}_i| + 1$ times it may be necessary to meet states from outside D_i.

This analysis demonstrates that T_2 contains $V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W$. Where d is less than n, the actual test may be much larger than this.

There are thus the following bounds.

- T_1 is contained in $V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W^C$.
- T_2 contains $V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W$.

While the upper bound on T_1 may be larger than the lower bound on T_2 there is little difference between them. The bounds are not tight. In particular, as $n - d$ grows T_2 is likely to become significantly larger than the lower bound given.

The following give sufficient conditions for T_2 to contain T_1.

Lemma 13 *Suppose W is a characterizing set for M. If W is a characterizing set for M^C then T_1 is contained in T_2.*

Proof

This follows from the fact that, as $W^C = W$, $V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W^C = V_{det}(\Sigma \cup \{\epsilon\})^{k+1}W$. □

Lemma 14 *Suppose V_{det} is a deterministic state cover for M and M is r-reduced. If V_{det} is a state cover for M^C then T_1 is contained in T_2.*

Proof

Suppose W is a characterizing set for M. As V_{det} is a state cover for M^C and M is r-reduced, W is a characterizing set for M^C. Thus $W^C = W$. The result thus follows from Lemma 13. □

The analysis provided here does not demonstrate that the approach given in Section 5 always generates a test set that is no larger than that generated by previous methods. It would be useful to have further necessary

and sufficient conditions for T_1 to be smaller than T_2 and for T_2 to be smaller than T_1. Tighter bounds might assist here. These conditions might allow the tester to choose the appropriate test generation algorithm for a particular problem. Note that there is an additional cost associated with adaptive testing: the cost of using a more complex test environment.

It may be useful to initially apply a breadth-first search using some set of input sequences W^H in order to distinguish states. This provides information about M_I that might greatly reduce the test effort. For example, such an initial phase might extend the state cover. Such a process might also provide information that can assist the tester in determining which test method should be used and can be achieved in a manner that is guaranteed not to increase the test effort.

7 Observations and future work

Future work will consider how some of the assumptions made may be weakened. In particular, it would be interesting to consider how the assumption that M is observable and completely specified may be weakened.

Section 5 described a general approach that produced a candidate DFSM M^C. The process used to produce M^C is adaptive and is based around a breadth-first search. A value of k is found and the state identification set $(\Sigma \cup \{\epsilon\})^k W$ initially used is based on this value of k. Instead, it is possible to use some smaller set W^H in order to look for new states of M^C. If new states are found, by being distinguished from those known, the value of k is reduced. This could, potentially, greatly reduce the test effort. Where W^H is contained within $(\Sigma \cup \{\epsilon\})^k W$, this does not increase the test effort even when no new states are found using W^H. This is because the tests executed in this initial search form part of the test set to be executed when generating the candidate.

The type of heuristic outlined above might be one of the great benefits of the approach given in this paper. Where many states are found by the initial search using W^H the value of k, and thus the test effort, is greatly reduced. It would be interesting to study heuristics that might drive such a process.

8 Conclusions

This paper has considered the problem of testing against a specification M that is a completely specified non-deterministic finite state machine (NFSM). It has been assumed that the implementation under test (IUT) I behaves like some unknown deterministic finite state machine M_I with at most m states for some known m.

Where the implementation is deterministic it is possible to use adaptive testing. In this paper, algorithms have been given to develop a candidate DFSM M^C. This candidate has the property that, under the test hypotheses used, I conforms to M if and only if M_I is equivalent to M^C. Where necessary, a test may then be generated from M^C using standard techniques.

The method outlined in this paper has a number of benefits. First, the test can be significantly shorter than those generated by previous methods. This is because the test is tailored to the IUT, rather than having to consider every implementation that satisfies the test hypotheses. It is also easier to generate a test from the candidate DFSM M^C than from the NFSM M and there are many more algorithms for generating tests from DFSMs than from NFSMs.

An interesting extension to this work would be the application of heuristics to the generation M^C. This may lead to M^C being generated using a significantly smaller test and thus reduce the test effort. It seems likely that there are heuristics that are capable of significantly reducing the test effort and that are guaranteed not to increase the test effort. This will be a topic for future work.

References

[1] S. Yu. Boroday. Distinguishing tests for non-deterministic finite state machines. In *Testing of Communicating Systems, IFIP TC6 11th International Workshop on Testing of Communicating Systems*, pages 101–107, Tomsk, Russia, August 31- September 2 1998. Kluwer Academic Press, Boston.

[2] T. S. Chow. Testing software design modelled by finite state machines. *IEEE Transactions on Software Engineering*, 4:178–187, 1978.

[3] D. I. Cohen. *Introduction to Computer Theory*. John Wiley and Sons, New York, second edition, 1997.

[4] M. C. Gaudel. Testing can be formal too. *Lecture Notes in Computer Science*, 915:82–96, 1995.

[5] A. Gill. *Introduction to The Theory of Finite State Machines*. McGraw-Hill, New York, 1962.

[6] D. Harel and M. Politi. *Modeling reactive systems with statecharts: the STATEMATE approach*. McGraw-Hill, New York, 1998.

[7] F. C. Hennie. Fault-detecting experiments for sequential circuits. In *Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 95–110, Princeton, New Jersey, November 1964.

[8] R. M. Hierons. Adaptive testing of a deterministic implementation against a nondetermistic finite state machine. *The Computer Journal*, 41:349–355, 1998.

[9] ITU-T. *Recommendation Z.500 Framework on formal methods in conformance testing*. International Telecommunications Union, Geneva, Switzerland, 1997.

[10] ITU-T. *Recommendation Z.100 Specification and description language (SDL)*. International Telecommunications Union, Geneva, Switzerland, 1999.

[11] V. Iyengar and K. Chakrabarty. An efficient finite-state machine implementation of Huffman decoders. *Information Processing Letters*, 64:271–275, 1998.

[12] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. *Synthesis of Finite State Machines: Functional Optimization*. Kluwer Academic Press, Boston, 1996.

[13] G. Luo, G. von Bochmann, and A. Petrenko. Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method. *IEEE Transactions on Software Engineering*, 20:149–162, 1994.

[14] A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das. Nondeterministic state machines in protocol conformance testing. In *Proceedings of Protocol Test Systems, VI (C-19)*, pages 363–378, Pau, France, 28-30 September 1994. Elsevier Science (North-Holland).

[15] A. Petrenko, N. Yevtushenko, and G. v. Bochmann. Testing deterministic implementations from nondeterministic FSM specifications. In *Testing of Communicating Systems, IFIP TC6 9th International Workshop on Testing of Communicating Systems*, pages 125–141, Darmstadt, Germany, 9-11 September 1996. Chapman and Hall, London.

[16] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.

[17] P. H. Starke. *Abstract Automata*. Elsevier, North-Holland, Amsterdam, 1972.

[18] A. S. Tanenbaum. *Computer Networks*. Prentice Hall International Editions, Prentice Hall, London, third edition, 1996.

[19] H. Ural, K. Saleh, and A. Williams. Test generation based on control and data dependencies within system specifications in SDL. *Computer Communications*, 23:609–627, 2000.

[20] H. Ural and K. Zhu. Optimal length test sequence generation using distinguishing sequences. *IEEE/ACM Transactions on Networking*, 1:358–371, 1993.

[21] M. P. Vasilevskii. Failure diagnosis of automata. *Cybernetics*, 4:653–665, 1973.