Testing from a Non-Deterministic Finite State Machine Using Adaptive State Counting

R. M. Hierons *Member, IEEE*

Abstract

The problem of generating a checking experiment from a non-deterministic finite state machine has been represented in terms of state counting. However, test techniques that use state counting traditionally produce preset test suites. This paper extends the notion of state counting in order to allow the input/output sequences observed in testing to be utilized: adaptive state counting is introduced. The main benefit of the proposed approach is that it may result in a reduction in the size of the test suite used. An additional benefit is that where a failure is observed it is possible to terminate test generation at this point.

Index Terms

Software Engineering, Software/Program Verification, Testing and Debugging, Non-deterministic finite state machine, Adaptive testing, State counting.

I. Introduction

MANY systems have some internal state that affects and is affected by operations of the system. Such systems, which include communications protocols and embedded control systems, are typically specified using state based languages such as Statecharts [6] and SDL [9]. Systems specified using these languages may be tested by applying methods based on *finite state machines (FSMs)*. A special type of FSM is a *deterministic finite state machine (DFSM)*. These test techniques are usually applied after the specification has been converted into an FSM by either expanding out the data (possibly after putting bounds on the types) or by applying some abstraction (see, for example, [10]).

The widespread use of state based systems, and the importance of their correctness, has led to much interest in testing from FSMs (see, for example, [1]–[3], [5], [7], [10]–[13], [16], [18]). Non-determinism in the specification is not unusual. Typically this comes either from some abstraction that has been applied or there being a number of acceptable output sequences in response to some input sequence. However, most work has focused on testing from DFSMs.

When testing from an FSM it is important to decide what is meant by correctness. This paper assumes that the implementation under test (IUT) is correct if and only if it is a reduction of the specification: every input/output sequence that is possible in the IUT is also present in the specification. This is an appropriate notion of correctness when the non-determinism in the specification is due to there being a set of alternative output sequences that are valid responses to some input sequence and the IUT may choose from these. An alternative is to test for equivalence: the IUT is deemed to be correct if and only if it is equivalent to the specification. Equivalence is the appropriate notion of correctness if all of the input/output sequences in the specification must be present in the IUT. Naturally these different notions of correctness lead to different test generation techniques but coincide where the specification is deterministic.

When testing from an FSM M it is normal to make certain assumptions and a checking experiment is a, typically preset, test suite that is guaranteed to determine correctness under these assumptions. Most approaches for generating a checking experiment from a non-deterministic FSM are based on the notion of state counting [12], [13], [18].

R.M. Hierons is with the Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

This paper introduces an (iterative) adaptive test generation algorithm: at each stage the algorithm produces the input sequences or adaptive test cases to be applied on the basis of the input/output sequences that have previously been observed. State counting is extended, to adaptive state counting, to allow observed input/output sequences to be utilized. This may reduce the size of the test suite used and the proposed test generation algorithm produces a test suite that determines whether the IUT is a reduction of the specification under the standard assumptions. The paper also formalizes the use of adaptive test cases [1], [16], which will be defined in Section IV, in conjunction with adaptive state counting. An additional benefit of adaptive state counting is that testing may be terminated if a failure is observed: where a preset test suite is used, the entire test suite is generated before testing proceeds. However, adaptive testing does require the use of a more sophisticated test environment.

This paper's main contributions are as follows. First, it explores properties of adaptive test cases. Second, it adapts the product machine of [13] to non-deterministic IUTs. It explores conditions under which the states of the IUT can be distinguished during testing. The paper then introduces adaptive state counting. An adaptive algorithm is given and we prove that this algorithm is correct. Finally, we prove that the test suite produced using the proposed algorithm is guaranteed to be contained within the test suite produced using state counting.

The paper is structured as follows. Section II introduces FSMs and Section III describes state counting. Section IV defines adaptive test cases and proves a number of properties of these. Section V adapts the product machine, that has been used in reasoning about testing a deterministic IUT against an FSM [13], to the case where the IUT may be non-deterministic. Section VI describes how states of the IUT may be distinguished during testing. This is followed, in Section VII, by a definition of adaptive state counting and an adaptive test generation algorithm. This algorithm is described in terms of the product machine. The proposed approach is evaluated in Section VIII and finally, in Section IX, conclusions are drawn.

II. Background

The testing of a state-based system using a preset test suite typically proceeds through the application of input sequences and the observation of the resultant output sequences. Suppose X denotes the set of inputs and Y denotes the set of outputs. An *input sequence* is a sequence $x_1, \ldots, x_k$ of inputs and an *input/output sequence* is a sequence $x_1/y_1, x_2/y_2, \ldots, x_k/y_k$ for some $x_1, \ldots, x_k \in X$ and $y_1, \ldots, y_k \in Y$. A *test sequence* is an input/output sequence $x_1/y_1, x_2/y_2, \ldots, x_k/y_k$ in which $y_1, \ldots, y_k$ is the specified response to $x_1, \ldots, x_k$. A *test suite* is a finite set of input sequences.

For convenience, an input/output sequence $\bar{a} = x_1/y_1, x_2/y_2, \ldots, x_k/y_k$ will sometimes be written $\bar{x}/\bar{y}$ where $\bar{x} = x_1, \ldots, x_k$ is the *input portion* of $\bar{a}$ and $\bar{y} = y_1, \ldots, y_k$ is the *output portion* of $\bar{a}$. Throughout this paper, any variable representing a sequence or tree will have a bar over its name.

An FSM M is defined by a tuple (S, s_1, X, Y, h) in which S is a finite set of states, $s_1 \in S$ is the initial state, X is the finite input alphabet, Y is the finite output alphabet, and h is the transition relation. The relation h has type $S \times X \leftrightarrow S \times Y$. Given state s and input x, $(s', y) \in h(s, x)$ if and only if the input of x when M is in state s may result in M moving to state s' and outputting y. The tuple $(s, s', x/y)$ defines a *transition* of M. The relation h may be extended to take input sequences. Consider, for example, the FSM M_0 described in Figure 1. Here $h(s_1, a) = \{(s_2, 0), (s_4, 1)\}$ and $h(s_1, bb) = \{(s_1, 10)\}$.

It is possible to define projections h^1 and h^2 of h such that h^1 gives the states reached from a state, given an input, and h^2 defines the input/output pairs from a state. These projections are defined by: $h^1(s, x) = \{s' \in S | \exists y \in Y.(s', y) \in h(s, x)\}$ and $h^2(s, x) = \{y \in Y | \exists s' \in S.(s', y) \in h(s, x)\}$. h^1 and h^2 may be extended to take input sequences. In M_0, $h^1(s_1, bb) = \{s_1\}$ and $h^2(s_1, bb) = \{10\}$.

The FSM $M = (S, s_1, X, Y, h)$ defines a language $L(M)$ which contains the input/output sequences allowed by M. More formally, $L(M) = \{\bar{x}/\bar{y} | \bar{x} \in X^* \wedge \bar{y} \in h^2(s_1, \bar{x})\}$. Similarly, the state s of M has an associated language: $L_M(s) = \{\bar{x}/\bar{y} | \bar{x} \in X^* \wedge \bar{y} \in h^2(s, \bar{x})\}$. Clearly $L(M) = L_M(s_1)$.

An FSM M is *completely specified* if for all $s \in S, x \in X$, $|h(s, x)| \geq 1$. If M is not completely specified it may be transformed to form a completely specified FSM. Three standard approaches for doing

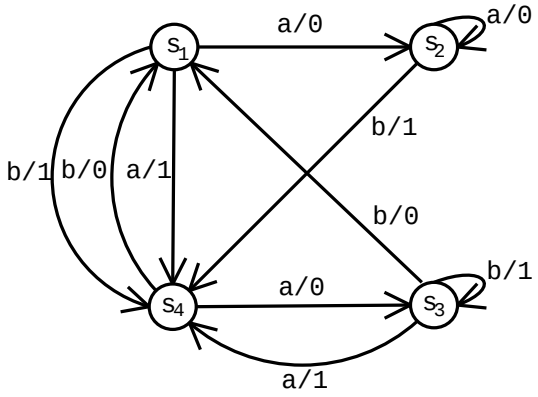

Fig. 1. The Non-deterministic Finite State Machine M_0

this are by adding an error state, a trap state, or self-loops with null output. M is *initially connected* if every state is reachable from the initial state of M: $\forall s \in S. \exists \bar{x} \in X^*. s \in h^1(s_1, \bar{x})$. If M is not initially connected it may be rewritten to form an initially connected FSM by removing the unreachable states. M has *reset capability* if it has a reset operation: some input r that takes every state to the initial state. The IUT has a *reliable reset* if it has a reset r that is known to have been implemented correctly. A reliable reset, that might be implemented through the system being switched off and then on again, may be used to separate input sequences. It will be assumed that any FSM considered is initially connected and completely specified and that the IUT has a reliable reset. The reliable reset will be represented by r and will not be included in the input alphabet X (it is treated differently in testing).

Two FSMs M_1 and M_2 are *equivalent* if and only if $L(M_1) = L(M_2)$. Two states s and s' of FSM M are *equivalent* if and only if $L_M(s) = L_M(s')$. An FSM M is *deterministic* if for every input sequence $\bar{x} \in X^*$ there is at most one output sequence $\bar{y} \in Y^*$ such that $\bar{x}/\bar{y} \in L(M)$. Note that in general it is not possible to convert an FSM into an equivalent DFSM. To see this, consider M_0. Here the input of a when M_0 is in state s_1 may lead to output 0 or 1 and so M_0 is not equivalent to a DFSM.

FSM M is said to be *observable* [14] if for every state s, input x, and output y, M has at most one transition leaving s with input x and output y. Every FSM is equivalent to an observable FSM [14]. It will thus be assumed that any FSM considered is observable. Given output sequence $\bar{y}$, $h^{\bar{y}}(s, \bar{x})$ will denote the state that is reached from s with input sequence $\bar{x}$ and output sequence $\bar{y}$: $\{h^{\bar{y}}(s, \bar{x})\} = \{s' \in S | (s', \bar{y}) \in h(s, \bar{x})\}$. If $\bar{y} \in h^2(s, \bar{x})$ then the set $\{s' \in S | (s', \bar{y}) \in h(s, \bar{x})\}$ is guaranteed to be a singleton because M is observable.

Recall that it is assumed that any FSM considered is completely specified. FSM M' is a *reduction* of FSM M if and only if M' has the same input alphabet as M and every input/output sequence that is possible in M' is allowed by M. More formally, an FSM M' is a *reduction* of FSM M if and only if M and M' have the same input alphabets and $L(M') \subseteq L(M)$. This is denoted $M' \preceq M$. Similarly, state s' of FSM M' is a *reduction* of state s of FSM M if and only if M and M' have the same input alphabets and $L_{M'}(s') \subseteq L_M(s)$. This is denoted $s' \preceq s$. This is similar to the notion of trace inclusion found in the labelled transition systems literature (see, for example, [15]).

In this paper we will assume that the IUT behaves like some unknown FSM M_I. The notion of correctness used is that the IUT is correct if and only if M_I is a reduction of the specification FSM M. This corresponds to the case in which, if the specification gives alternative output sequences in response to some input sequence $\bar{x}$, these output sequences are acceptable alternatives. By contrast, where correctness is equivalence, if the specification gives alternative output sequences in response to some input sequence $\bar{x}$, a correct IUT must be capable of producing all of these alternatives.

The above notions may be generalized in the following way. State s' of M' is a reduction of state s of M on test suite D if and only if M and M' have the same input alphabets and every input/output sequence produced from s' with an input sequence in D is allowed from s. More formally, state s' of M'

is a reduction of state s of M on test suite D if and only if M and M' have the same input alphabets and $\{\bar{x}/\bar{y} \in L_{M'}(s') | \bar{x} \in D\} \subseteq \{\bar{x}/\bar{y} \in L_M(s) | \bar{x} \in D\}$. This is denoted $s' \preceq_D s$ and otherwise $s' \not\preceq_D s$. Suppose s_1 is the initial state of M and s'_1 is the initial state of M'. Then M' is a reduction of M on D if and only if $s'_1 \preceq_D s_1$. This is denoted $M' \preceq_D M$ and otherwise $M' \not\preceq_D M$.

When testing from an FSM it is usual to assume that the IUT behaves likes some unknown element of a fault domain: the set Ψ_M^m of completely specified observable FSMs with the same input and output alphabets[1] as M and at most m states (some predetermined m). A test suite is called a *checking experiment* if and only if for every $M' \in \Psi_M^m$, that is not a reduction of M, the test suite shows that M' is erroneous[2]. More formally, D is a *checking experiment* if and only if for all $M' \in \Psi_M^m$, $M' \preceq M \Leftrightarrow M' \preceq_D M$. Throughout this paper it will be assumed that the IUT behaves like some unknown observable FSM $M_I = (T, t_1, X, Y, h_I) \in \Psi_M^m$.

When testing a non-deterministic implementation it is normal to make a fairness assumption, sometimes called the *complete testing assumption*, that there is some known k such that if an input sequence is applied k times then all possible responses are observed (see, for example, [11]). This paper will assume that such a fairness assumption can be made. Naturally, this assumption holds immediately in the important case where the implementation is known to be deterministic.

III. Applying state counting

This section will briefly review the literature on testing from FSMs, concentrating on the use of state counting. It is organized as follows. Section III-A describes the notion of a deterministic state cover. Section III-B considers how states of an FSM may be distinguished. Section III-C then describes the use of state counting in generating a checking experiment. When the implementation is known to be deterministic, this knowledge may be used in testing [7], [13]. Future work will consider how the results in this paper may be strengthened where it is known that the IUT is deterministic.

A. Reaching states of the specification

Input sequence $\bar{x} \in X^*$ is said to *deterministically-reach (d-reach)* state s if and only if $h^1(s_1, \bar{x}) = \{s\}$: s is the only state reached by $\bar{x}$. s is then said to be *d-reachable*. For example, in M_0 s_4 is d-reached by b and thus is d-reachable. By contrast, s_2 is not d-reachable.

If $\bar{x}$ d-reaches s and M_I is a reduction of M then each state of M_I that may be reached by input sequence $\bar{x}$ must be a reduction of s. A set V of input sequences is a *deterministic state cover* if it contains the empty sequence ϵ and is a minimal set such that every d-reachable state s of M is d-reached by some input sequence from V [13]. S_V denotes the set of d-reachable states of M. $V = \{\epsilon, b, ba\}$ is a deterministic state cover for M_0.

A test suite will be produced by extending sequences from V. While V need not reach all of the states of either the specification or the IUT, reasoning based on adaptive state counting will be used in order to determine when it is possible to stop extending the test suite.

B. Distinguishing states of the specification

When testing from an FSM M it is useful to have sequences that distinguish states of M. In order for an input sequence $\bar{x}$ to distinguish two states s and s' of M it is sufficient that the corresponding sets of output sequences do not intersect. More formally, this is if $h^2(s, \bar{x}) \cap h^2(s', \bar{x}) = \emptyset$ [14]. This notion of distinguishing states may be extended in the following, intrinsically adaptive, way [1], [13].

Definition 1: States s and s' are r(1)-distinguishable if there is some input $x \in X$ such that $h^2(s, x) \cap h^2(s', x) = \emptyset$. States s and s' are r(k)-distinguishable ($k > 1$) if either s and s' are r(j)-distinguishable for

[1]There may be outputs with the property that it appears to be feasible that the IUT can produce these even though the specification FSM cannot. Where this is the case, we will assume that Y has been extended to include these outputs.

[2]Checking experiments may be defined similarly for other fault domains [3].

TABLE I

THE POSSIBLE RESPONSES OF M_0 TO W

State	Responses to aa	Responses to ba
s_1	$00, 10$	10
s_2	00	10
s_3	10	$00, 01, 11$
s_4	01	$00, 01$

some $1 \leq j < k$ or there is some input $x \in X$ such that for all $y \in h^2(s,x) \cap h^2(s',x)$ the states $h^y(s,x)$ and $h^y(s',x)$ are r(j)-distinguishable for some $1 \leq j < k$. States s and s' are *r-distinguishable* if there is some $k \geq 1$ such that s and s' are r(k)-distinguishable.

Given r-distinguishable states s and s' it is possible that there is no single input sequence that r-distinguishes them. The notion of r-distinguishing states leads to the use of a set of input sequences $W(s,s')$, called an r-distinguishing set [13], to r-distinguish states s and s'. A set W' of input sequences *r-distinguishes* states s and s' if W' contains some r-distinguishing set for s and s'. The set $W(s,s')$ can be defined inductively [13].

Definition 2: A set W of input sequences is a *characterizing set* if it r-distinguishes each pair of r-distinguishable states of M.

Proposition 1: Given states s and s' of M, if $L_M(s') \subseteq L_M(s)$ then s and s' are not r-distinguishable

Now consider the example M_0. By Proposition 1, since $L_{M_0}(s_2) \subseteq L_{M_0}(s_1)$ we know that s_1 and s_2 are not r-distinguishable. Table I shows that the set $W = \{aa, ba\}$ r-distinguishes all other states and so is a characterizing set.

C. State Counting

This section describes state counting and its use in generating a checking experiment from an FSM. The problem is to determine, through black-box testing, whether the IUT may exhibit an input/output sequence that is not in the language $L(M)$ defined by the specification.

The test suite will be developed using a breadth-first search through input sequences. In order to apply a search it is necessary to have some termination criterion that decides whether an input sequence needs to be extended. Recall that I behaves like some unknown $M_I = (T, t_1, X, Y, h_I) \in \Psi_M^m$. Given an observed input/output sequence in $L(M)$, we may consider the current (unknown) state t of M_I and the current state s of M. A failure occurs in response to the next input if and only if the input/output exhibited from t is not allowed from s. Thus a failure is associated with a pair $(s,t) \in S \times T$ of states.

A termination criterion for the search will be based on the observation that if a state pair $(s,t) \in S \times T$, from which a failure may be exhibited, is reachable then it is reachable by some *minimal length* input/output sequence $\bar{x}/\bar{y}$. If a prefix $\bar{x}_1/\bar{y}_1$ of $\bar{x}/\bar{y}$ reaches state pair (s',t') then $\bar{x}_1/\bar{y}_1$ must define a minimal sequence to (s',t'). Thus, if it is possible to demonstrate that a sequence reaches some such pair of states that has already been met then this input/output sequence need not be extended since it cannot form the prefix of a minimal sequence to a failure. State counting is used to demonstrate this: the reasoning used is based on placing a lower bound on the number of separate states of M_I that must have been visited if there has been no repetition in the pairs of states met. Since M_I has at most m states, once this lower bound exceeds m the sequence must have repeated a pair of states and so the sequence need not be extended. When all output sequences observed in response to an input sequence $\bar{x}$ have this property, $\bar{x}$ need not be extended further.

We will briefly describe test generation based on state counting[3]. Let $S_1, \ldots, S_z$ denote maximal sets of r-distinguishable states of M. Given $S' \subseteq S$, $\hat{S}'$ will denote the set of states from S' that are d-reachable: $\hat{S}' = S' \cap S_V$. W will denote the characterizing set used. Given a d-reachable state $s \in S_V$, a set $Tr(s)$ (called a traversal set in [12]) is constructed in the following way:

[3]Adaptive state counting, which is based on related observations, will be described in depth in Section VII.

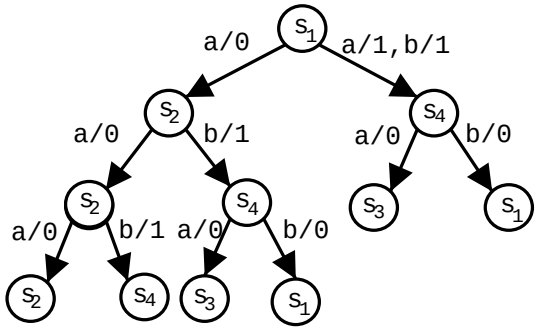

Fig. 2. The tree representing F_1

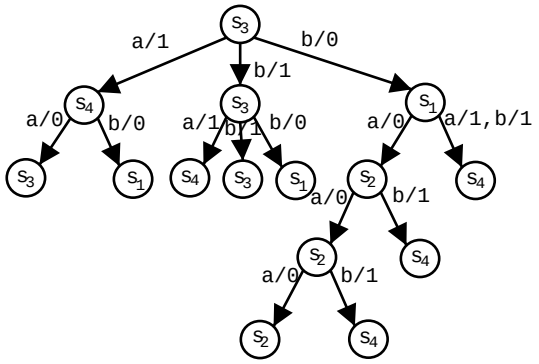

Fig. 3. The tree representing F_3

- On the basis of the successor tree, generate a set $F_s \subseteq L_M(s)$ of input/output sequences such that: for each input/output sequence $\bar{x}/\bar{y} \in F_s$ there is some S_i, $1 \leq i \leq z$, such that $\bar{x}/\bar{y}$ visits states from S_i exactly $m - |\hat{S}_i| + 1$ times when followed from s and this condition does not hold for any proper prefix of $\bar{x}/\bar{y}$.
- $Tr(s)$ is the set of input sequences such that there is some corresponding input/output sequence in F_s: $Tr(s) = \{\bar{x} \in X^* | \exists \bar{y} \in Y^*.\bar{x}/\bar{y} \in F_s\}$.

Given a set $A \subseteq X^*$, let $\mathcal{T}(\bar{v}_i, A)$ denote the set of input sequences formed by following $\bar{v}_i$ by each prefix of a sequence in A. More formally, $\mathcal{T}(\bar{v}_i, A)$ is the set $\{\bar{v}_i\}Pre(A)$, where $Pre(A)$ denotes the set of prefixes of sequences from A (i.e. $Pre(A) = \cup_{\bar{a} \in A} pre(\bar{a})$ where $pre(\bar{a}) = \{\bar{a}_1 | \exists \bar{a}_2.\bar{a} = \bar{a}_1\bar{a}_2\}$). The following test suite is produced [11]:

$$E = \bigcup_{s_i \in S_V} \mathcal{T}(v_i, Tr(s_i))W$$

Now consider the application of state counting to the example FSM M_0 with $m = n = 4$. Here the deterministic state cover V reaches states s_1, s_3, and s_4. Further, the characterizing set $W = \{aa, ba\}$ distinguishes all of the states except s_1 and s_2. There are thus two maximal sets of r-distinguishable states: $S_0 = \{s_1, s_3, s_4\}$ and $S_1 = \{s_2, s_3, s_4\}$. Here $\hat{S}_0 = S_0$ and thus $|\hat{S}_0| = 3$. $\hat{S}_1 = \{s_3, s_4\}$ and so $|\hat{S}_1| = 2$. Thus, a node in the successor tree is a leaf if one of the following holds:

- After the root, on the path to the leaf there are at least two nodes that represent states from S_0.
- After the root, on the path to the leaf there are at least three nodes that represent states from S_1.

This leads to the sets F_1, F_3, and F_4 represented by the trees in Figures 2, 3, and 4 respectively.

Recall, that the set F_i defines the set of input sequences produced by taking the prefixes of the set of paths from the root to a leaf. Since $W = \{aa, ba\}$, the tree F_1 leads to the following test suite:

$$\{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb\}\{aa, ba\}$$

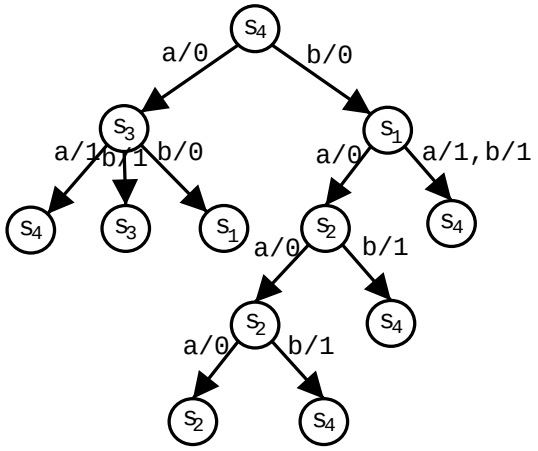

Fig. 4. The tree representing F_4

The tree F_3 leads to the test suite:

$$\{ba\}\{\epsilon, a, b, aa, ab, ba, bb, baa, bab, baaa, baab\}\{aa, ba\}$$

The tree F_4 leads to the test suite:

$$\{b\}\{\epsilon, a, b, aa, ab, ba, bb, baa, bab, baaa, baab\}\{aa, ba\}$$

The complete test suite is produced by taking the union of these three sets. The following result is from Luo et al. [11].

Theorem 2: The set E of input sequences is a checking experiment.

We will now introduce new notation that will be used to rephrase state counting. This will make it easier to compare the test suites produced by state counting and adaptive state counting.

Given input sequence $\bar{x} \in X^*$ there may be a number of alternative output sequences that may be produced in response to $\bar{x}$ and some of these might satisfy the termination criterion while others do not. Thus, it is possible for there to be two input sequences in $Tr(s)$ such that one is a proper prefix of the other. An input sequence $\bar{x}$ in $Tr(s)$ is a maximal element of $Tr(s)$ if for every output sequence $\bar{y} \in h^2(s, \bar{x})$, some prefix of $\bar{x}/\bar{y}$ is in F_s. The notion of an input sequence being a maximal element of $Tr(s)$ will be represented in terms of $LB_{sc}(s, S_1, \bar{x})$.

$$LB_{sc}(s, S_1, \bar{x}) = min_{\bar{y} \in h^2(s,\bar{x})} |\{\bar{x}'/\bar{y}' \in pre(\bar{x}/\bar{y}) \setminus \{\epsilon\} | h^{\bar{y}'}(s, \bar{x}') \in S_1\}| + |\hat{S}_1|$$

$|\{\bar{x}'/\bar{y}' \in pre(\bar{x}/\bar{y}) \setminus \{\epsilon\} | h^{\bar{y}'}(s, \bar{x}') \in S_1\}|$ is the number of times $\bar{x}/\bar{y}$ visits states from S_1, when followed from s. Thus, $LB_{sc}(s, S_1, \bar{x})$ counts the number of times states from S_1 are visited by $\bar{x}/\bar{y}$ and V for *each* output sequence $\bar{y} \in h^2(s, \bar{x})$ and takes the *minimum* of these values. If this reaches $m + 1$ then the input sequence $\bar{x}$ need not be further extended: for every $\bar{y} \in h^2(s, \bar{x})$, some prefix of $\bar{x}/\bar{y}$ is in $Tr(s)$. Thus $\bar{x}$ is a maximal input sequence in $Tr(s)$.

Proposition 3: An input sequence $\bar{x}$ is a maximal input sequence in $Tr(s)$ if and only if there exists a set S_1 of r-distinguishable states of M such that $LB_{sc}(s, S_1, \bar{x}) = m + 1$ and for every set S'_1 of r-distinguishable states of M, $LB_{sc}(s, S'_1, \bar{x}) \leq m + 1$.

Proposition 4: Suppose $\bar{v}_i, \bar{v}_j \in V$ are prefixes of $\bar{x}$ that reach states s_i and s_j respectively, $\bar{x} = \bar{v}_i\bar{x}_i$, and $\bar{x} = \bar{v}_j\bar{x}_j$. Let S_1 denote some set of r-distinguishable states. If $\bar{v}_i$ is a prefix of $\bar{v}_j$ then $LB_{sc}(s_i, S_1, \bar{x}_i) \geq LB_{sc}(s_j, S_1, \bar{x}_j)$.

Test generation using state counting may thus be rephrased in the following way.

Algorithm 1: 1) Set $\mathcal{T} = V$ and $\mathcal{T}_C = V$.

2) While $\mathcal{T}_C \neq \emptyset$

3) For every input sequence $\bar{x} \in \mathcal{T}_C$, do the following:

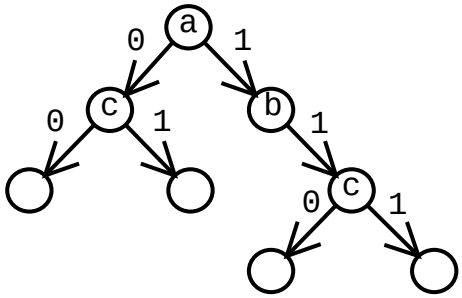

Fig. 5. An adaptive test case

a) Find the maximal element $\bar{v}$ of V that is a prefix of $\bar{x}$.
b) Find $\bar{x}'$ and s such that $\bar{x} = \bar{v}\bar{x}'$ and $\bar{v}$ d-reaches s.
c) Remove $\bar{x}$ from $\mathcal{T}_C$ if there is some set S_1 of r-distinguishable states of M with $LB_{sc}(s, S_1, \bar{x}') > m$.

4) Set $\mathcal{T}_C = \mathcal{T}_C X \setminus \mathcal{T}$ and $\mathcal{T} = \mathcal{T} \cup \mathcal{T}_C X$
5) endwhile
6) Output the test suite $\mathcal{T}W$.

In the algorithm the set $\mathcal{T}_C$ is the set of input sequences currently being considered in the search and $\mathcal{T}$ is the set of input sequences considered to date. If input sequence $\bar{x} \in \mathcal{T}_C$ satisfies the termination criterion it is removed from $\mathcal{T}_C$ in step 3. Otherwise $\bar{x}$ is extended in step 4. When $\bar{x}$ is extended, it is sufficient to consider extensions to $\bar{x}$ that have yet to be considered (which is why the set $\mathcal{T}$ is removed from $\mathcal{T}_C X$ when extending $\mathcal{T}_C$).

This algorithm extends input sequences until they satisfy the termination criterion. One possible termination criterion is to insist that for each $\bar{v}_i$ that is a prefix of $\bar{x}$, it is not necessary to extend $\bar{x}$ when considering the corresponding F_j. However, according to Proposition 4, it is sufficient to consider only the maximal prefix of $\bar{x}$ that is contained in V and this is the approach used in Algorithm 1.

When all the states of M are d-reachable and r-distinguishable, the test suite reduces to the set $V(X \cup \{\epsilon\})^{m-n+1}W = V(\{\epsilon\} \cup X \cup \ldots \cup X^{m-n+1})W$. This is equivalent to the test produced, using the W-method [4], [17], when testing from a DFSM. Where these conditions do not hold, a larger test suite is required.

The use of state counting when testing a deterministic IUT against an FSM has been described in terms of the product machine [13]. Section V will adapt the product machine to the case where the implementation may be non-deterministic. Before this, adaptive test cases will be explored.

IV. Adaptive test cases

This section introduces the notion of an adaptive test case. It then formalizes this idea and proves results that will be used later. Informally an adaptive test case is a rooted tree with directed edges. In this tree, each leaf represents the adaptive test case terminating and every other node has an associated input. The edges represent outputs and there cannot be more than one edge with output y leaving a node n. Figure 5 represents an adaptive test case in which a, b, and c are inputs and 0 and 1 are outputs.

An adaptive test case is applied in the following manner. We start at the root. Suppose we have reached node n. If n is a leaf we stop. Otherwise, if n has input x then we apply x and observe the output y produced. If there is no edge from n with output y, we terminate; otherwise we move to the node n' reached by the edge from n with label y. For example, in applying the adaptive test case in Figure 5, we first input a. If 0 is output we then input c. We then terminate, irrespective of the next output produced.

It is natural to define trees recursively. In doing so, a node n can have one of two forms: it can be a leaf (represented by $null$) or it has two components: an input x and a set of pointers to nodes (roots of subtrees), one pointer for each edge from n. This set of pointers, to nodes, can be represented by a partial function f: if there is an edge, with output y, from n to some node n' then $f(y)$ is the adaptive test case

represented by n'. The set Υ of *all* adaptive test cases, with input alphabet X and output alphabet Y, may be defined recursively [8].

Definition 3: Υ is the set of adaptive test cases, where an adaptive test case $\bar{\sigma} \in \Upsilon$ is one of:

- $null$
- a pair (x, f) in which x is an input and f is a partial function from output values to adaptive test cases. Thus, f is a partial function from Y to Υ.

An adaptive test case $\bar{\sigma} \in \Upsilon$ is applied in the following manner. If $\bar{\sigma} = null$ then the adaptive test case ends. If $\bar{\sigma} = (x, f)$ then the input x is applied and some output y is observed. If f is not defined on y we terminate and otherwise we apply the adaptive test case $f(y)$. It will be assumed that any adaptive test case considered is finite: its application must always terminate. The function f is partial in order to allow a more concise description of adaptive test cases in which, at some nodes, certain output values are known to indicate a failure and thus to lead to no further input.

Consider the adaptive test case in Figure 5. Here the root node is (a, f_1) for a function f_1 in which $f_1(0)$ is the node (c, f_2) and $f_1(1) = (b, f_3)$. The function f_2 is defined by $f_2(0) = null$ and $f_2(1) = null$ while the function f_3 is defined by $f_3(1) = (c, f_4)$. Finally, $f_4(0) = null$ and $f_4(1) = null$.

An input sequence may be seen as an adaptive test case in which the functions represent constants: the next input applied is the same irrespective of the output. Thus the results that will be developed for adaptive test cases apply when using input sequences. Given input sequence $\bar{x}$ and adaptive test case $\bar{\sigma}$, it is possible to follow $\bar{x}$ by $\bar{\sigma}$: we simply apply the input sequence $\bar{x}$ to the IUT, observe the resultant output sequence and then apply the adaptive test case $\bar{\sigma}$.

Given adaptive test case $\bar{\sigma}$, the length of $\bar{\sigma}$ is the length of the longest input/output sequence that may result from the application of $\bar{\sigma}$.

Definition 4: The *length* of an adaptive test case $\bar{\sigma}$, $length(\bar{\sigma})$, is [8]:

- 0 if $\bar{\sigma} = null$
- $1 + max\{length(f(y)) | y \in dom\ f \wedge y \in Y\}$ if $\bar{\sigma} = (x, f)$

where $dom\ f$ denotes the elements of Y on which f is defined.

Consider, for example, the adaptive test case $\bar{\sigma}$ in Figure 5. Here $length(\bar{\sigma}) = 1 + max\{length((c, f_2)), length((b, f_3))\}$. $length((c, f_2)) = 1 + max\{length(null), length(null)\} = 1$. $length((b, f_3)) = 1 + max\{length(null), length((c, f_4))\}$ but since $length((c, f_4)) = 1 + max\{length(null), length(null)\} = 1$, $length((b, f_3)) = 2$. Thus, $length(\bar{\sigma}) = 1 + max\{1, 2\} = 3$.

AboElFotoh et al. [1] discuss the use of adaptive test cases to distinguish states. A similar notion is described by Tripathy and Naik [16]. AboElFotoh et al. give algorithms for generating adaptive test cases that distinguish states. The notions that lie behind the use of adaptive test cases will now be formalized.

Given adaptive test case $\bar{\sigma}$ and state u of FSM $N = (U, u_1, X, Y, h_N)$, $IO_N(u, \bar{\sigma})$ will denote the set of input/output sequences that may be observed by applying $\bar{\sigma}$ to N when N is in state u. $IO_N(u, \bar{\sigma})$ is:

$$\{\epsilon\} \text{ if } \bar{\sigma} = null$$

$$\left(\bigcup_{y \in h_N^2(u,x) \wedge y \notin dom\ f} \{x/y\} \right) \cup \left(\bigcup_{y \in h_N^2(u,x) \wedge y \in dom\ f} \{x/y\} IO_N(h_N^y(u, x), f(y)) \right) \text{ if } \bar{\sigma} = (x, f)$$

The first rule states that if the adaptive test case is $null$ then, since no input is applied, the empty sequence is observed. The second rule is recursive, stating that if the input of x may lead to output y ($y \in h_N^2(u, x)$) then $\bar{\sigma}$ may lead to an input/output sequence in the form of x/y followed by either termination (if $y \notin dom\ f$ and so f does not define a next input) or some input/output sequence formed by applying $f(y)$ in the state $h_N^y(u, x)$ reached from u by x/y. Each input/output sequence in $IO_N(u, \bar{\sigma})$ is a *possible response* to $\bar{\sigma}$ when N is in state u and $IO_N(u, \bar{\sigma})$ is the *set of responses* of N to $\bar{\sigma}$ when in state u. Consider the example M_0 and the adaptive test case $\bar{\sigma}_1$ in Figure 6. Then $IO_{M_0}(s_1, \bar{\sigma}_1) = \{aa/00, a/1\}$ and $IO_{M_0}(s_3, \bar{\sigma}_1) = \{a/1\}$.

This notation may be extended to sets of adaptive test cases. Given set Σ of adaptive test cases and state u of FSM $N = (U, u_1, X, Y, h_N)$, $IO_N(u, \Sigma)$ is defined by the following.

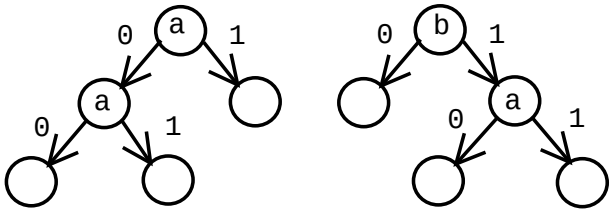

Fig. 6. The adaptive test cases $\bar{\sigma}_1$ and $\bar{\sigma}_2$

$$IO_N(u, \Sigma) = \bigcup_{\bar{\sigma} \in \Sigma} IO_N(u, \bar{\sigma})$$

The notion of an adaptive test case (x, f) r-distinguishing two states s and s' of M is quite natural: the possible responses to (x, f) in states s and s' should be disjoint. This is the case if and only if, where s and s' may both lead to some output y in response to x ($y \in h^2(s, x) \cap h^2(s', x)$), the remaining adaptive test case must be guaranteed to r-distinguish the states reached from s and s' by x/y. Clearly, if $h^2(s, x) \cap h^2(s', x) = \emptyset$ then (x, f) r-distinguishes s and s' for any choice of f.

Definition 5: An adaptive test case $\bar{\sigma} = (x, f)$ ***r-distinguishes*** states s and s' of M if and only if for all $y \in h^2(s, x) \cap h^2(s', x)$, we have that $y \in dom\, f$ and $f(y)$ r-distinguishes the states $h^y(s, x)$ and $h^y(s', x)$.

Consider the example M_0. Here the set $\{aa, ba\}$ is a characterizing set. However, if aa is input and the first output is 1 then the second output does not help distinguish the states. Similarly, when considering ba, if the response to b is 0 then there is no need to apply a. Thus the r-distinguishable states of M_0 are r-distinguished by the adaptive test cases $\bar{\sigma}_1$ and $\bar{\sigma}_2$ shown in Figure 6.

The following result relates the approaches of using an adaptive test case to r-distinguish two states and the corresponding sets of input/output sequences.

Lemma 5: Adaptive test case $\bar{\sigma} \in \Upsilon$ r-distinguishes states s and s' of M if and only if $IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma}) = \emptyset$.

Proof: Case 1: $\Rightarrow$. Proof by induction on the length of $\bar{\sigma}$. The base case, with length 0, follows immediately. Inductive hypothesis: for every adaptive test case $\bar{\sigma}' \in \Upsilon$ of length less than p, $p > 0$, if $\bar{\sigma}'$ r-distinguishes states s and s' of M then $IO_M(s, \bar{\sigma}') \cap IO_M(s', \bar{\sigma}') = \emptyset$. Suppose $\bar{\sigma} = (x, f) \in \Upsilon$ has length p and r-distinguishes s and s'.

Proof by contradiction: suppose $IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma}) \neq \emptyset$. Let $x\bar{x}_1/y\bar{y}_1$ be some element of $IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma})$ ($x \in X$ and $y \in Y$). Thus $y \in h^2(s, x) \cap h^2(s', x)$. Let $s_0 = h^y(s, x)$ and $s'_0 = h^y(s', x)$. By the definition of IO_M and the observability of M, $\bar{x}_1/\bar{y}_1 \in IO_M(s_0, f(y)) \cap IO_M(s'_0, f(y))$.

By definition, since $\bar{\sigma}$ r-distinguishes s and s', we know that $y \in dom\, f$ and $f(y)$ r-distinguishes s_0 and s'_0. Further, $f(y)$ has length at most $p - 1$. Thus, by the inductive hypothesis, $IO_M(s_0, f(y)) \cap IO_M(s'_0, f(y)) = \emptyset$. This provides a contradiction as required.

Case 2: $\Leftarrow$. Proof by induction on the length of $\bar{\sigma}$. The base case, with length 0, follows immediately. Inductive hypothesis: for every $\bar{\sigma}' \in \Upsilon$ of length less than p, $p > 0$, if $IO_M(s, \bar{\sigma}') \cap IO_M(s', \bar{\sigma}') = \emptyset$ then $\bar{\sigma}'$ r-distinguishes s and s'. Suppose $\bar{\sigma} = (x, f) \in \Upsilon$ has length p and $IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma}) = \emptyset$.

It is sufficient to prove that for all $y \in h^2(s, x) \cap h^2(s', x)$, we have that $y \in dom\, f$ and $f(y)$ r-distinguishes states $h^y(s, x)$ and $h^y(s', x)$. Suppose $y \in h^2(s, x) \cap h^2(s', x)$ and let $s_0 = h^y(s, x)$ and $s'_0 = h^y(s', x)$. As $IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma}) = \emptyset$ we must have that $y \in dom\, f$. Observe that since $IO_M(s, \sigma) \cap IO_M(s', \bar{\sigma}) = \emptyset$, $IO_M(s_0, f(y)) \cap IO_M(s'_0, f(y)) = \emptyset$. Further, $length(f(y)) < p$. Thus, by the inductive hypothesis, $f(y)$ r-distinguishes s_0 and s'_0. The result thus follows. ■

It is now possible to introduce notation regarding the use of adaptive test cases to r-distinguish states.

Definition 6: Given adaptive test case $\bar{\sigma} \in \Upsilon$, a set Σ of adaptive test cases, completely specified FSM $M = (S, s_1, X, Y, h)$, and completely specified FSM $M_I = (T, t_1, X, Y, h_I)$:

- State t of M_I is a reduction of state s of M on $\bar{\sigma}$ if and only if $IO_{M_I}(t,\bar{\sigma}) \subseteq IO_M(s,\bar{\sigma})$. This is denoted $t \preceq_{\bar{\sigma}} s$.
- State t of M_I is a reduction of state s of M on Σ if and only if t is a reduction of state s on every element of Σ. This is denoted $t \preceq_{\Sigma} s$.
- M_I is a reduction of M on $\bar{\sigma}$ if and only if $t_1 \preceq_{\bar{\sigma}} s_1$. This is written $M_I \preceq_{\bar{\sigma}} M$.
- M_I is a reduction of M on Σ if and only if $t_1 \preceq_{\Sigma} s_1$. This is written $M_I \preceq_{\Sigma} M$.

The following definition extends the notion of a characterizing set to adaptive test cases.

Definition 7: A set Ω of adaptive test cases is an *adaptive characterizing set* for M if and only if for all $s, s' \in S$, if s and s' are r-distinguishable then they are r-distinguished by some element of Ω.

V. The Product Machine

The problem of testing a deterministic implementation against an FSM has been described in terms of the product machine [13]. The state of the product machine is either a special state $Fail$ or a pair $(s,t) \in S \times T$ of states that represent the states of M and $M_I \in \Psi_M^m$ given the input/output sequence observed. The product machine behaves like M_I where this is consistent with M and otherwise moves to the state $Fail$. Naturally, since M_I is unknown before testing the product machine is also unknown. However, testing may be seen as trying to decide whether the state $Fail$ of the (unknown) product machine is reachable and this observation helps when reasoning about the effectiveness of a test suite.

This section adapts the definition of the product machine to the case where the implementation may be non-deterministic. Given observable FSM $M = (S, s_1, X, Y, h)$ and observable FSM $M_I = (T, t_1, X, Y, h_I)$ that models the IUT, the product machine $P(M, M_I) = (S \times T \cup \{Fail\}, (s_1, t_1), X, Y \cup \{fail\}, h_p)$ where for all $x \in X$, $h_p(Fail, x) = \{(Fail, fail)\}$ and for all $x \in X$, $(s,t) \in S \times T$, and $y \in Y$:

1) If $(t', y) \in h_I(t, x)$ and $(s', y) \in h(s, x)$ then $((s', t'), y) \in h_P((s,t), x)$.
2) If $(t', y) \in h_I(t, x)$ and $y \notin h^2(s, x)$ then $(Fail, y) \in h_P((s,t), x)$.

Lemma 6: $P(M, M_I)$ is observable.

Proof: This is an immediate consequence of the fact that M and M_I are observable. ■

Note that, if incorrect output can be produced by the IUT in response to an input sequence then the 'first incorrect output' of the IUT is produced by the product machine (from the corresponding state). Only after this is 'fail' produced. This differs slightly from the previous definition [13] in which this 'first incorrect output' is not produced by the product machine. The following results show that the problem of deciding whether the IUT is correct is equivalent to deciding whether $Fail$ is reachable.

Lemma 7: Let $\bar{x}/\bar{y}$ denote an input/output sequence ($\bar{x} \in X^*, \bar{y} \in Y^*$). Then $Fail = h_P^{\bar{y}}((s_1, t_1), \bar{x})$ if and only if there exists some prefix $\bar{x}'/\bar{y}'$ of $\bar{x}/\bar{y}$ with $\bar{x}'/\bar{y}' \in L(M_I) \setminus L(M)$.

Proof: Case 1: $\Leftarrow$. Proof by contradiction: suppose $\bar{x}'/\bar{y}' \in L(M_I) \setminus L(M)$ and $Fail \neq h_P^{\bar{y}}((s_1, t_1), \bar{x})$. Then $Fail \neq h_P^{\bar{y}'}((s_1, t_1), \bar{x}')$. Let $(s,t) = h_P^{\bar{y}'}((s_1, t_1), \bar{x}')$. Clearly $\bar{y}'$ does not contain the element $fail$. By the definition of the product machine, since $\bar{x}'/\bar{y}'$ reaches state $(s,t) \neq Fail$, $s \in h^{\bar{y}'}(s_1, \bar{x}')$. This contradicts $\bar{x}'/\bar{y}' \notin L(M)$ as required.

Case 2: $\Rightarrow$. Suppose $Fail = h_P^{\bar{y}}((s_1, t_1), \bar{x})$. Let $\bar{x}'/\bar{y}'$ denote some minimal prefix of $\bar{x}/\bar{y}$ that reaches $Fail$. By the definition of the product machine, $\bar{x}'/\bar{y}' \in L(M_I)$. Thus it is sufficient to prove that $\bar{y}' \notin h^2(s_1, \bar{x}')$. $\bar{x}'/\bar{y}' = \bar{x}_1 x_2/\bar{y}_1 y_2$ for some $\bar{x}_1 \in X^*$, $\bar{y}_1 \in Y^*$, $x_2 \in X$, and $y_2 \in Y$.

Since, by Lemma 6, the product machine is observable, $h_P^{\bar{y}_1}((s_1, t_1), \bar{x}_1)$ is defined. By the minimality of $\bar{x}'/\bar{y}'$, $\bar{x}_1/\bar{y}_1$ reaches some state $(s,t) = h_P^{\bar{y}_1}((s_1, t_1), \bar{x}_1)$ other than $Fail$ of $P(M, M_I)$. Observe now that $Fail = h_P^{y_2}((s,t), x_2)$. Thus, by the definition of the product machine, $y_2 \notin h^2(s, x_2)$. Since M is observable, s is the only state of M reached by input/output sequence $\bar{x}_1/\bar{y}_1$. Thus $\bar{y}_1 y_2 \notin h^2(s_1, \bar{x}_1 x_2)$ and so $\bar{y}' \notin h^2(s_1, \bar{x}')$ as required. ■

Theorem 8: Suppose that the IUT I, that behaves like an unknown element $M_I \in \Psi_M^m$, is being tested against the FSM M. Then M_I is a reduction of M if and only if the state $Fail$ of $P(M, M_I)$ is not reachable from the initial state of $P(M, M_I)$.

Proof: This follows immediately from Lemma 7. ■

Deciding correctness is now expressed in terms of deciding reachability for the (unknown) product machine. Section VII will define adaptive state counting and explain how it may be used to construct a test suite that determines this reachability. Adaptive state counting will rely on distinguishing states of the IUT during testing and this will be described in Section VI.

VI. Distinguishing states of the implementation

Each adaptive test case will be repeated a sufficient number of times for us to assume, under fairness, that all possible responses of M_I have been observed. Thus $\bar{\sigma}$ distinguishes two states t and t' of M_I if the set of possible input/output sequences observed by applying $\bar{\sigma}$ in t and t' differ. This observation motivates the following definition of what it means to distinguish two states of the IUT.

Definition 8: An adaptive test case $\bar{\sigma} \in \Upsilon$ *distinguishes* states t and t' of M_I if and only if $IO_{M_I}(t, \bar{\sigma}) \neq IO_{M_I}(t', \bar{\sigma})$. If some adaptive test case from Σ distinguishes t and t' we say that Σ *distinguishes* t and t'.

The notion of distinguishing states of the implementation in this manner will prove to be useful when applying adaptive testing. The following shows that if a set Σ of adaptive test cases r-distinguishes states s and s' of M and states t, t' of M_I satisfy $t \preceq_\Sigma s$ and $t' \preceq_\Sigma s'$ then Σ distinguishes t and t'.

Theorem 9: Suppose that $\Sigma \subseteq \Upsilon$ r-distinguishes states s and s' of M and states t, t' of M_I satisfy $t \preceq_\Sigma s$ and $t' \preceq_\Sigma s'$. Then Σ distinguishes t and t'.

Proof: Since Σ r-distinguishes s and s', there exists some adaptive test case $\bar{\sigma} \in \Sigma$, $\bar{\sigma} \neq null$, that r-distinguishes s and s'. By Lemma 5, $IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma}) = \emptyset$. Since $t \preceq_\Sigma s$ and $t' \preceq_\Sigma s'$, $IO_{M_I}(t, \bar{\sigma}) \subseteq IO_M(s, \bar{\sigma})$ and $IO_{M_I}(t', \bar{\sigma}) \subseteq IO_M(s', \bar{\sigma})$. Thus $IO_{M_I}(t, \bar{\sigma}) \cap IO_{M_I}(t', \bar{\sigma}) \subseteq IO_M(s, \bar{\sigma}) \cap IO_M(s', \bar{\sigma}) = \emptyset$. Since M_I is completely specified and $\bar{\sigma} \neq null$, $IO_{M_I}(t, \bar{\sigma}) \neq \emptyset$. Thus, $IO_{M_I}(t, \bar{\sigma}) \neq IO_{M_I}(t', \bar{\sigma})$ and the result follows. ■

VII. Adaptive State Counting

Throughout this section Ω will denote the adaptive characterizing set used. Since a characterizing set defines an adaptive characterizing set, the results and techniques in this section extend immediately to the use of a characterizing set to r-distinguish states.

Adaptive state counting will proceed in a manner similar to state counting: we start with V and keep on extending input sequences (followed by Ω) until a termination criterion is satisfied. Given an input/output sequence $\bar{x}/\bar{y}$, the termination criterion will be based on finding some number j such that if $\bar{x}/\bar{y}$ does not repeat a state of the product machine then M_I must have at least j states. The contribution of Ω is that it distinguishes some states of M_I and, in particular, if $t \preceq s$ and $t' \preceq s'$ ($t, t' \in T$, $s, s' \in S$) and Ω r-distinguishes s and s' then Ω must distinguish t and t'. An input sequence does not have to be extended if $j > m$ for every output sequence, since it cannot be a prefix of some minimal sequence to a failure. The key difference is that, since the algorithm is adaptive, in calculating j we have additional information: observed input/output sequences.

We get a number of benefits from adaptivity. Recall that in calculating $LB_{sc}(s, S_1, \bar{x})$, in order to decide whether a sequence $\bar{x}$ must be extended, we take a *minimum* over all $\bar{y} \in h^2(s, \bar{x})$. If certain input/output sequences that are contained in the specification are not contained in the IUT then we do not need to consider these sequences in deciding whether $\bar{x}$ should be extended. This may lead to earlier termination. Further, Ω might distinguish two states of the IUT reached by certain input/output sequences even if Ω does not distinguish the corresponding states of the specification. Both of these advantages can be used in calculating j and thus lead to a reduction in the size of the test suite used. Finally, if a failure is observed we can terminate without creating the rest of the test suite.

Sufficient repetitions will be used so that it can be assumed, under fairness, that all possible responses have been observed. Section VIII will briefly discuss how the fairness assumption may be extended to the use of adaptive test cases. Before describing adaptive state counting, a number of terms will be defined.

A. Characterizing the states reached by a sequence

Given input/output sequence $\bar{x}/\bar{y}$ observed in testing, $B_\Omega(\bar{x}/\bar{y})$ will denote the set of all input/output sequences that may be produced by M_I if we apply elements of Ω in the state of M_I reached by $\bar{x}/\bar{y}$. Thus $B_\Omega(\bar{x}/\bar{y}) = IO_{M_I}(t, \Omega)$ where $t = h_I^{\bar{y}}(t_1, \bar{x})$ is the state of M_I reached by $\bar{x}/\bar{y}$. By fairness, all of these input/output sequences will be observed in testing if $\bar{x}/\bar{y}$ is followed by Ω. Thus if two input/output sequences lead to states of M_I that are distinguished by Ω then they lead to *different* states of M_I.

Proposition 10: If $B_\Omega(\bar{x}/\bar{y}) \neq B_\Omega(\bar{x}'/\bar{y}')$ then $h_I^{\bar{y}}(t_1, \bar{x}) \neq h_I^{\bar{y}'}(t_1, \bar{x}')$.

Suppose that $\bar{v}/\bar{v}'$ is an input/output sequence of M_I that may be observed in response to some $\bar{v} \in V$. It will be useful to consider the states of M_I that may be reached using prefixes of some sequence $\bar{x}/\bar{y}$ following $\bar{v}/\bar{v}'$. Note that M_I and M may allow more than one response to $\bar{v}$ and these input/output sequences may reach different states of M_I even though they reach the same state of M.

Given $s' \in S$, if $\bar{v}\bar{x}/\bar{v}'\bar{y}$ is an input/output sequence that can be produced by both M and M_I ($\bar{v}\bar{x}/\bar{v}'\bar{y} \in L(M_I) \cap L(M)$), $R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ will denote prefixes of $\bar{v}\bar{x}/\bar{v}'\bar{y}$ that reach s' (in M) and that extend $\bar{v}/\bar{v}'$.

$$R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}) = \{\bar{v}\bar{x}'/\bar{v}'\bar{y}'|\bar{x}'/\bar{y}' \in pre(\bar{x}/\bar{y}) \setminus \{\epsilon\} \wedge s' = h^{\bar{v}'\bar{y}'}(s_1, \bar{v}\bar{x}')\}$$

When considering an input/output sequence $\bar{v}\bar{x}/\bar{v}'\bar{y} \in L(M) \cap L(M_I)$, if this does not repeat states of the product machine then its prefixes that are in $R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ must reach $|R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})|$ *different* states of M_I since each of the input/output sequences reaches the same state of M.

Proposition 11: Suppose $\bar{v} \in V$, $\bar{v}\bar{x}/\bar{v}'\bar{y} \in L(M) \cap L(M_I)$, $s' \in S$, and no state of the product machine has been repeated when (in testing) $\bar{v}/\bar{v}'$ is followed by $\bar{x}/\bar{y}$. Then the states of M_I reached by input/output sequences in $R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ are distinct.

Given set $\mathcal{T}$ of input sequences, $B_\Omega(\mathcal{T})$ denotes the set of responses to Ω that may be observed from states of the IUT reached by $\mathcal{T}$:

$$B_\Omega(\mathcal{T}) = \{B_\Omega(\bar{x}/\bar{y})|\bar{x} \in \mathcal{T} \wedge \bar{x}/\bar{y} \in L(M_I)\}$$

.

Each element of $B_\Omega(\mathcal{T})$ is a distinct set of input/output sequences produced in response to Ω and must represent *at least* one state of M_I.

B. A lower bound

This section will introduce a lower bound that may be placed on the number of states of M_I if there has been no repetition in states of the product machine for a given input/output sequence. This will drive adaptive state counting: whenever this lower bound exceeds m for every observed response to an input sequence $\bar{x}$, we know that it is not necessary to extend $\bar{x}$ since $\bar{x}$ cannot be a prefix of a minimal input sequence that can lead to failure.

Before defining the lower bound, we will consider the states of M_I reached by sequences from V. Let $V = \{\bar{v}_1, \ldots, \bar{v}_p\}$. For each $\bar{v}_i \in V$, the set V_i' will denote the set of possible responses of the IUT to $\bar{v}_i$: $V_i' = h_I^2(t_1, \bar{v}_i)$. Each element $\bar{v}_i' \in V_i'$ may correspond to a different state of M_I: the state $h_I^{\bar{v}_i'}(t_1, \bar{v}_i)$. The lower bound will consider the set V', defined below. V' represents the set of possible ways of choosing individual elements from each V_i'.

$$V' = \{\{\bar{v}_1/\bar{v}_1', \bar{v}_2/\bar{v}_2', \ldots, \bar{v}_i/\bar{v}_i', \ldots \bar{v}_p/\bar{v}_p'\}|\forall 1 \leq j \leq p.\bar{v}_j' \in V_j'\}$$

Note that since V must contain ϵ, each element of V' contains ϵ/ϵ. In testing, every input sequence in the deterministic state cover V will be followed by the adaptive characterizing set. This motivates the introduction of new notation. Given $V'' \in V'$, $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ is formed by taking the set $R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ (of input/output sequences of the form $\bar{v}\bar{x}'/\bar{v}'\bar{y}'$ that are prefixes of $\bar{v}\bar{x}/\bar{v}'\bar{y}$ and reach s' in M) and adding the input/output sequence from V'' that reaches s' in M, if there is such a sequence.

Suppose $s, s' \in S$, $s = h^{\bar{v}'}(s_1, \bar{v})$, and $V'' \in V'$. Then $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ is defined by the following.

1) If s' is d-reached by some $\bar{v}_1 \in V$ and $\bar{v}_1/\bar{v}'_1 \in V''$ then
$$R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'') = R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}) \cup \{\bar{v}_1/\bar{v}'_1\}$$
2) Otherwise
$$R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'') = R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$$

All the input/output sequences in $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ reach s' in M. Thus, if no state of the product machine is repeated, the states of M_I reached by the input/output sequences in $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ must be distinct.

We now have the components that will contribute to adaptive state counting. Suppose that $\bar{v} \in V$, $\bar{v}\bar{x}/\bar{v}'\bar{y} \in L(M) \cap L(M_I)$, $S_1 \subseteq S$, Ω is the adaptive characterizing set used, $V'' \in V'$, and $\bar{v}/\bar{v}' \in V''$. Further, suppose $\mathcal{T}$ denotes the set of input sequences that have been followed by Ω in testing. In Lemma 12, we will prove a property of the term $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'')$, defined below, that will be used in adaptive state counting. This term is defined by the sum of two parts which will now be explained.

1) The first part is $\sum_{s' \in S_1} |R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')| = \sum_{s' \in S_1} |R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})| + |\hat{S}_1|$. Each of the sequences in $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ reaches the same state (s') of M and thus, if no state of the product machine is repeated then the input/output sequences in $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ must reach *different* states of M_I.
   Suppose that for all $s, s' \in S_1$ such that $s \neq s'$, we have that Ω distinguishes every state of M_I reached by an input/output sequence in $R^+(s, \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ from every state of M_I reached by an input/output sequence in $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$. Note that this condition is automatic if the states in S_1 are r-distinguished by Ω and no failures are observed. If this condition holds, the set of states of M_I reached by input/output sequences in $R^+(s, \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ is disjoint from the set of states of M_I reached by input/output sequences in $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$.
   Under these conditions, by Proposition 11, the input/output sequences in the $R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ meet $\sum_{s' \in S_1} |R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')|$ distinct states of M_I.
2) The second part is $|B_\Omega(\mathcal{T}) \setminus (\cup_{s' \in S_1, \bar{x}_1/\bar{y}_1 \in R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')} B_\Omega(\bar{x}_1/\bar{y}_1))|$. This is the number of sets of responses to Ω that have been observed from states of M_I and that have not been observed from states considered in the previous term. By Proposition 10, each of these sets of responses must correspond to an additional state of M_I.

The term $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'')$ is defined by:

$$LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'') = \sum_{s' \in S_1} |R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})| + |\hat{S}_1| +$$
$$|B_\Omega(\mathcal{T}) \setminus (\bigcup_{s' \in S_1, \bar{x}_1/\bar{y}_1 \in R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')} B_\Omega(\bar{x}_1/\bar{y}_1))|$$

The third term in this expression denotes the number of additional sets of input/output sequences observed in response to Ω. Each of these must correspond to a state of the IUT.

Lemma 12: Suppose that

1) $\bar{v} \in V$, $\bar{v}\bar{x}/\bar{v}'\bar{y} \in L(M) \cap L(M_I)$, and $\bar{v}/\bar{v}'$ is the maximal length prefix of $\bar{v}\bar{x}/\bar{v}'\bar{y}$ in V''.
2) $\mathcal{T}$ denotes the total set of input sequences that have been followed by Ω in testing and there have been sufficient repetitions so that under fairness we can assume that all possible responses have been observed.
3) $\mathcal{T}$ contains every sequence in V and every sequence of the form $\bar{v}\bar{x}'$ for a prefix $\bar{x}'$ of $\bar{x}$.
4) $S_1 \subseteq S$ has the property that for all $s_1, s_2 \in S_1$, $s_1 \neq s_2$, Ω distinguishes every state of M_I reached by an input/output sequence in $R^+(s_1, \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ from every state of M_I reached by an input/output sequence in $R^+(s_2, \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$.
5) No failures are observed.

If no state (s,t) of the product machine reached by a sequence $\bar{v}\bar{x}_0/\bar{v}'\bar{y}_0$ ($\bar{x}_0/\bar{y}_0$ is a non-empty prefix of $\bar{x}/\bar{y}$) is reached by some $\bar{v}\bar{x}'/\bar{v}'\bar{y}'$ for a prefix $\bar{x}'/\bar{y}' \neq \bar{x}_0/\bar{y}_0$ of $\bar{x}/\bar{y}$ or by some input/output sequence in V'' then M_I must have at least $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'')$ states.

Proof: First observe that given $s_i, s_j \in S_1$, $s_i \neq s_j$, each state reached by a sequence in $R(s_i, \bar{v}/\bar{v}', \bar{x}/\bar{y})$ is distinguished by Ω from each state reached by a sequence in $R(s_j, \bar{v}/\bar{v}', \bar{x}/\bar{y})$. Further, since no state of the product machine is repeated along $\bar{x}/\bar{y}$ from $\bar{v}/\bar{v}'$, by Proposition 11 the sequences in some $R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ ($s' \in S_1$) must reach different states of M_I. Thus, the sequences in $\cup_{s' \in S_1} R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ must reach different states of M_I. The sequences in each $R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})$ ($s' \in S_1$) must also reach states that are not reached by sequences in V''. Thus the sequences in $(\cup_{s' \in S_1} R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})) \cup V''$ reach $\sum_{s' \in S_1} |R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})| + |\hat{S}_1|$ different states of M_I.

By Proposition 10, every set of responses to Ω must indicate a state of M_I. Thus, M_I must have at least $|B_\Omega(\mathcal{T}) \setminus (\bigcup_{s' \in S_1, \bar{x}_1/\bar{y}_1 \in R^+(s', \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')} B_\Omega(\bar{v}/\bar{v}', \bar{x}_1/\bar{y}_1))|$ additional states. The result thus follows. ■

This result will drive adaptive state counting. Given input sequence $\bar{v}\bar{x}$ used in testing, we extend $\bar{v}\bar{x}$ if it might form the prefix of a minimal sequence to a failure. For this to be the case we must have some response $\bar{v}'\bar{y}$ to $\bar{v}\bar{x}$ such that $\bar{v}\bar{x}/\bar{v}'\bar{y}$ that does not repeat a state of the product machine. This corresponds to the last part of the statement of Lemma 12. By choosing appropriate S_1 and V'', we can ensure that the other conditions of Lemma 12 hold and thus, if $\bar{v}\bar{x}/\bar{v}\bar{y}$ does not repeat any state of the product machine then M_I must have *at least* $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'')$ states. This provides a contradiction if $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'') > m$, in which case $\bar{v}\bar{x}/\bar{v}'\bar{y}$ must repeat a state of the product machine and thus need not be extended.

C. Adaptive state counting: an algorithm

The following is a test generation algorithm based on adaptive state counting. In this algorithm $\mathcal{T}$ denotes the set of input sequences that have been followed by Ω in testing. $\mathcal{T}_C$ denotes the set of current elements of $\mathcal{T}$: those that are being considered in the search through the state space of the product machine. The elements in $\mathcal{T}_C$ are the maximal sequences considered that do not meet the termination criterion. On each iteration, elements of $\mathcal{T}_C$ are either removed from $\mathcal{T}_C$ or extended.

Algorithm 2:
1) Set $\mathcal{T} = V$ and $\mathcal{T}_C = V$.
2) While $\mathcal{T}_C \neq \emptyset$
3) Test the IUT a sufficient number of times, in order to be able to apply the fairness assumption, with each element of $\mathcal{T}_C\Omega$ and record the set of input/output sequences observed in response to the input sequences in $\mathcal{T}_C$ and the corresponding set of responses to Ω. If a failure is observed, output the set of input/output sequences that have been observed and terminate.
4) For each input sequence $\bar{x}_1 \in \mathcal{T}_C$, remove $\bar{x}_1$ from $\mathcal{T}_C$ if for every response $\bar{y}_1$ to $\bar{x}_1$ observed there exists $S_1 \subseteq S$ and $V'' \in V'$ such that the following hold:
   a) $\bar{x}_1/\bar{y}_1 = \bar{v}\bar{x}/\bar{v}'\bar{y}$, where $\bar{v}/\bar{v}'$ is the maximal element of V'' that is a prefix of $\bar{x}_1/\bar{y}_1$;
   b) For all $s_1, s_2 \in S_1$ with $s_1 \neq s_2$, Ω distinguishes every state of M_I reached by an input/output sequence from $R^+(s_1, \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$ from every state of M_I reached by an input/output sequence from $R^+(s_2, \bar{v}/\bar{v}', \bar{x}/\bar{y}, V'')$; and
   c) $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'') > m$.
5) Set $\mathcal{T}_C = \mathcal{T}_C X \setminus \mathcal{T}$ and $\mathcal{T} = \mathcal{T} \cup \mathcal{T}_C X$.
6) endwhile
7) Output the set of input/output sequences that have been observed and the fact that the IUT passed the test suite applied.

In deciding whether the termination condition holds, in principle all subsets of S and all elements of V' must be considered. Naturally this may not be practical. One way of choosing S_1 is to start with the maximal sets of r-distinguishable states of M and extend these. It will transpire that even if we restrict

ourselves to the maximal sets of r-distinguishable states of M, then for *any* choice of $V'' \in V'$ the lower bound produced here is no less than that produced with state counting and may be larger.

Theorem 13: Suppose the IUT I behaves like an observable FSM with the same input and output alphabets as M and with at most m states. Algorithm 2 states that I passes the test suite applied if and only if I is a reduction of M.

Proof: Case 1: $\Rightarrow$. This follows from Lemma 12 and the fact that the input sequences followed by Ω are extended until the termination criterion is satisfied.

Case 2: $\Leftarrow$. This follows from I being equivalent to some $M_I \in \Psi_M^m$ and from the definition of M_I being a reduction of M. ■

VIII. Evaluation

This section will evaluate adaptive state counting by comparing it to state counting. First, the fairness assumption is discussed. Section VIII-B contains a proof that the test suite produced using adaptive state counting is contained within that produced using state counting and contains further general observations. Finally, in Section VIII-C, adaptive state counting is applied to the example.

A. The fairness assumption

Where the IUT is known to be deterministic, the fairness assumption automatically holds. Further, if a characterizing set, rather than an adaptive characterizing set, is used in adaptive state counting then the normal fairness assumption can be made. This section will now briefly consider how a fairness assumption might be applied when using an adaptive characterizing set. A fuller analysis of this issue will be left to future work.

Given adaptive test case $\bar{\sigma}$, let $W_{\bar{\sigma}}$ denote the set of maximal input sequences that may result from the application of $\bar{\sigma}$. These are the maximal input sequences that label paths from the root to some leaf of the tree corresponding to $\bar{\sigma}$. For example, in the adaptive test case $\bar{\sigma}$ in Figure 5 , $W_{\bar{\sigma}} = \{ac, abc\}$.

Theorem 14: If, under fairness, it is sufficient to apply any input sequences k times then it is also sufficient to apply each adaptive test case k times.

Proof: Consider some adaptive test case $\bar{\sigma}$ and the corresponding set $W_{\bar{\sigma}} = \{\bar{x}_1, \ldots, \bar{x}_p\}$. Suppose $\bar{x}/\bar{y}$ is a possible response of the IUT to $\bar{\sigma}$. Then $\bar{x}$ is a prefix of $\bar{x}_i$ for some $1 \leq i \leq p$.

Under fairness, if we apply $\bar{x}_i$ k times, we are guaranteed to see every possible response of the IUT to $\bar{x}_i$. Suppose $\bar{y}_i$ is one of these possible responses such that $\bar{y}$ is a prefix of $\bar{y}_i$. We may now observe that if in an execution the IUT responds to $\bar{x}_i$ to produce $\bar{y}_i$ then it would have produced $\bar{x}/\bar{y}$ in response to $\bar{\sigma}$. Thus, if we apply $\bar{\sigma}$ k times we are guaranteed to observe $\bar{x}/\bar{y}$. From this we may conclude that by applying $\bar{\sigma}$ k times we are guaranteed to observe all possible responses to $\bar{\sigma}$. ■

B. General Results

This section explores properties of adaptive state counting. First, we show that the test suite produced by adaptive state counting is contained within that produced by state counting. The following shows that adaptive state counting terminates the extension of input sequences no later than state counting.

Theorem 15: Suppose that $V'' \in V'$, $\bar{v}/\bar{v}' \in V''$, $\bar{v}\bar{x}/\bar{v}'\bar{y} \in L(M) \cap L(M_I)$, and $\mathcal{T}$ denotes the set of input sequences followed by Ω in Algorithm 2. Suppose S_1 is some set of r-distinguishable states of M and Algorithm 2 does not observe any failures. Then $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'') \geq LB_{sc}(s, S_1, \bar{x})$.

Proof: Recall that

$$LB_{sc}(s, S_1, \bar{x}) = min_{\bar{y} \in h^2(s,\bar{x})} |\{\bar{x}'/\bar{y}' \in pre(\bar{x}/\bar{y}) \setminus \{\epsilon\} | h^{\bar{y}'}(s, \bar{x}') \in S_1\}| + |\hat{S}_1|$$

$$LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'') \geq \sum_{s' \in S_1} |R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})| + |\hat{S}_1|$$

Observe that

$$\sum_{s' \in S_1} |R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})| = |\{\bar{x}'/\bar{y}' \in pre(\bar{x}/\bar{y}) \setminus \{\epsilon\} | h^{\bar{y}'}(s, \bar{x}') \in S_1\}|$$

The result now follows. ■

By considering the example, it will be demonstrated that $LB(v/v', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'')$ may be strictly greater than $LB_{sc}(s, S_1, \bar{x})$.

Theorem 16: The test suite applied using adaptive state counting is contained in that produced using state counting.

Proof: First note that if a failure is observed in adaptive state counting then Algorithm 2 terminates. By contrast, state counting produces a preset test suite. Thus, it is sufficient to consider the case where no failures are observed during the application of Algorithm 2.

Suppose adaptive characterizing set Ω is being used, S_1 is a set of r-distinguishable states of M, $s_1, s_2 \in S_1$, $s_1 \neq s_2$, for states t_1 and t_2 of M_I we have $t_1 \preceq_\Omega s_1$ and $t_2 \preceq_\Omega s_2$, and no failures are observed in the application of Algorithm 2. By Theorem 9, t_1 and t_2 are distinguished by Ω. Thus when the set S_1 may be used in state counting in order to show that a sequence need not be extended, S_1 can also be used in adaptive state counting. The result now follows from Theorem 15. ■

The following gives a condition under which the test suite generated using adaptive state counting and characterizing set W is guaranteed to be contained within that produced if the W-method is applied.

Proposition 17: Suppose that M has n states, the IUT behaves like some FSM $M_I \in \Psi_M^m$, and the deterministic state cover V reaches each state of M. Suppose that for every pair $\bar{x}_1, \bar{x}_2$ of input sequences, with $(t^i, \bar{y}_i) \in h_I(t_1, \bar{x}_i)$ and $s^i = h^{\bar{y}_i}(s_1, \bar{x}_i)$ ($i \in \{1, 2\}$), if $s^1 \neq s^2$ then t^1 and t^2 are distinguished by W. Then the test suite produced using adaptive state counting is contained in the set $V(X \cup \{\epsilon\})^{m-n+1}W$.

Proof: Observe that, under the conditions, if two input/output sequences from $L(M_I)$ reach different states of M then the corresponding states of M_I are distinguished by W. Thus, in calculating the lower bound it is possible to choose $S_1 = S$. Given this choice of S_1, $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'') \geq \sum_{s' \in S} |R(s', \bar{v}/\bar{v}', \bar{x}/\bar{y})| + |\hat{S}| = |\bar{x}/\bar{y}| + n$. Thus, $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S, \Omega, V'') > m$ if $|\bar{x}/\bar{y}| \geq m - n + 1$. The result thus follows. ■

Now consider the expected reduction in the size of the test suite. In the worst case, the test suite will be the same as that produced using state counting: observing the response of the IUT provides no additional useful information. Suppose a state of the implementation, not reached by V'', is distinguished from those met by V''. Then, for some sequences this will increase the last term, in $LB(\bar{v}/\bar{v}', \bar{x}/\bar{y}, \mathcal{T}, S_1, \Omega, V'')$, by one. This will lead to a number of sequences terminating one step earlier and thus may lead to a reduction of the order of $|X|$ in the size of the test suite. Thus, where j extra states are found, the size of the test suite may be reduced by the order of $|X|^j$. Naturally, the actual reduction will depend upon a number of properties of the specification and implementation.

C. Applying adaptive state counting to the example

In order to illustrate the potential savings, suppose that the IUT behaves like the FSM M_I^1 in Figure 7, M_0 is the specification, and characterizing set $\Omega = \{aa, ba\}$ is used (instead of an adaptive characterizing set). The first iteration of the algorithm uses the test suite $V\Omega = \{\epsilon, b, ba\}\{aa, ba\}$. This identifies three responses to Ω and thus three separate states of M_I^1. None of the sequences used satisfies the termination criterion and thus all are extended.

The second iteration uses the test suite $VX\Omega = \{\epsilon, b, ba\}\{a, b\}\{aa, ba\}$. We observe a fourth response to Ω: that of the state reached by $a/0$. Thus, 4 separate states of M_I^1 have been found. All of the sequences, that do not pass through this state, have the third term in the lower bound taking on the value 1. Based on this, it is straightforward to show that all of the sequences except $\bar{v}_0\{a\} = \{a\}$ are leaves as they give a lower bound of 5 using $S_1 = S$ ($|\hat{S}| = 3$). We now need only extend the sequence a to get $\{aa, ab\}\{aa, ba\}$: the two nodes reached are leaves. Thus the following test suite was used:

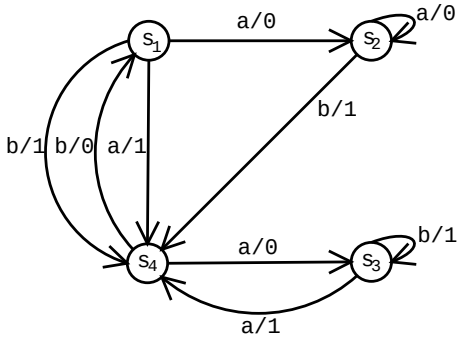

Fig. 7. The FSM M_I^1

$$\{\epsilon, a, b, aa, ab, ba, bb, baa, bab\}\{aa, ba\}$$

Suppose each input has cost 1 and each input sequence ends with a reset with cost 1. The test suite has cost 86. This contrasts with the test suite produced using state counting which has cost 342. Note that the above test suite may be further reduced by observing in advance that the set $VX\Omega$ must be used and thus by removing every sequence in $V\Omega \cup VX\Omega$ that is a prefix of some other sequence in $V\Omega \cup VX\Omega$. In this case, the sequences in $V\Omega$ are all prefixes of sequences in $VX\Omega$. Further, the sequences $baaa$ and $baba$ from $VX\Omega$ are prefixes of other sequences from $VX\Omega$. This observation leads to the test suite being reduced to one with total length 62. In contrast, once prefixes are removed, state counting leads to a test suite with cost 201. This gives a 69% reduction.

Interestingly, this illustrates a potential weakness of applying an adaptive test generation algorithm: it is not always possible to remove an input sequence $\bar{x}$ that is a prefix of another input sequence $\bar{x}'$ from the test suite since when $\bar{x}$ is input it may not be known that $\bar{x}'$ will be used. This happened in the above case: the input sequences aaa and aba in $VX\Omega$ are prefixes of sequences in $\{a\}X\Omega$. Future work will consider heuristics that might maximize the potential of saving through the removal of prefixes. One simple heuristic operates as follows. First generate test suite $\mathcal{T}_1$ using state counting. In adaptive state counting, when considering input sequence $\bar{x}$ apply some maximal input sequence $\bar{x}'$ from $\mathcal{T}_1$ such that $\bar{x}$ is a prefix of $\bar{x}'$. This increases the potential for savings through prefix removal and guarantees that the test suite is contained within that produced by state counting.

In this case the test suite may be further reduced by using an adaptive characterizing set. When applying aa, if the first output is 1 then the second input need not be applied. This occurs from s_3 and in one response to a from s_1. When applying ba, if the first output is 0 the second input need not be applied.

While adaptive state counting may lead to significantly smaller test suites, there are other aspects to the costs of testing. In particular, adaptive testing requires a more complex test environment.

IX. Conclusion

This paper has considered the problem of utilizing adaptivity when testing against a non-deterministic finite state machine (FSM). Two forms of adaptivity have been considered: the use of (preset) adaptive test cases to distinguish states and the use of information derived during testing to drive the generation of a test suite. The latter leads to an adaptive algorithm, in which input sequences are applied and then further input sequences are generated on the basis of the input/output sequences that have been observed.

It has been shown that testing may be based around adaptive state counting which is an extension of the notion of state counting [12], [13], [18]. It has been proved that the use of adaptive state counting is guaranteed to produce a test suite that is contained within that produced by state counting. Further reductions may result from using adaptive test cases, to distinguish states, rather than input sequences.

By contrast with state counting, adaptive state counting allows properties of the IUT discovered during testing to be utilized. It has been shown that this is capable of leading to a significant reduction in the size of the test suite.

Future work will consider how, when using adaptive state counting, the test suite may be further reduced where the implementation is known to be deterministic. It will also consider how the assumptions, about the specification, may be relaxed.

ACKNOWLEDGMENT

This work was partially funded by EPSRC grant GR/R43150 Formal Methods and Testing (FORTEST).

REFERENCES

[1] H. AboElFotoh, O. Abou-Rabia, and H. Ural. A test generation algorithm for protocols modeled as non-deterministic FSMs. *The Software Engineering Journal*, 8(4):184–188, 1993.

[2] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *27th ACM Symposium on Theory of Computing*, pages 363–372, 1995.

[3] 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.

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

[5] S. Fujiwara and G. v. Bochmann. Testing non-deterministic state machines with fault coverage. In *Proceedings of Protocol Test Systems, IV*, pages 267–280, 1991.

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

[7] R. M. Hierons. Adaptive testing of a deterministic implementation against a nondetermistic finite state machine. *The Computer Journal*, 41(5):349–355, 1998.

[8] R. M. Hierons and H. Ural. Concerning the ordering of adaptive test sequences. In *23rd IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2003), LNCS volume 2767*, pages 289–302, Berlin, Germany, September 29 - October 2 2003. Springer-Verlag.

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

[10] D. Lee and M. Yannakakis. Principles and methods of testing finite-state machines - a survey. *Proceedings of the IEEE*, 84(8):1089–1123, 1996.

[11] G. L. Luo, G. v. Bochmann, and A. Petrenko. Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method. *IEEE Transactions on Software Engineering*, 20(2):149–161, 1994.

[12] 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).

[13] 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.

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

[15] J. Tretmans. Conformance testing with labelled transitions systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29(1):49–79, 1996.

[16] P. Tripathy and K. Naik. Generation of adaptive test cases from non-deterministic finite state models. In *Proceedings of the 5th International Workshop on Protocol Test Systems*, pages 309–320, Montreal, September 1992.

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

[18] N. V. Yevtushenko, A. V. Lebedev, and A. F. Petrenko. On checking experiments with nondeterministic automata. *Automatic Control and Computer Sciences*, 6:81–85, 1991.