Mutation Testing from Probabilistic Finite State Machines*

Robert M. Hierons
Brunel University,
Uxbridge, Middlesex,
United Kingdom, UB8 3PH

Mercedes G. Merayo
Brunel University and
Universidad Complutense de Madrid,
28040 Madrid, Spain

Abstract

Mutation testing traditionally involves mutating a program in order to produce a set of mutants and using these mutants in order to either estimate the effectiveness of a test suite or to drive test generation. Recently, however, this approach has been applied to specifications such as those written as finite state machines. This paper extends mutation testing to finite state machine models in which transitions have associated probabilities. The paper describes several ways of mutating a probabilistic finite state machine (PFSM) and shows how test sequences that distinguish between a PFSM and its mutants can be generated. Testing then involves applying each test sequence multiple times, observing the resultant output sequences and using results from statistical sampling theory in order to compare the observed frequency of each output sequence with that expected.

1 Introduction

In recent years we have observed an evolution in the kind of systems and properties that formal methods are dealing with. In the beginning they mainly concentrated on the functional behaviour of systems, that is, on what a system could/should do. This led to formalisms such as the (original) notions of process algebras, Petri nets, and Moore/Mealy machines. The next step was to deal with quantitative information such as the *probabilities* resolving the non-deterministic choices that a system may undertake (see, for example, [24, 16, 7, 27]). There exist several areas where probabilistic representations have been applied successfully including computational linguistics, pattern recognition, bio-informatics and circuit testing.

In order to specify systems dealing with probabilities we will use *probabilistic finite state machines (PFSMs)* that are finite state machines with probabilities attached to their transitions. Intuitively, a transition in a finite state machine indicates that if the machine is in a state s and receives an input i then it can produce an output o and change its state to s'. An appropriate notation for such a transition could be $s \xrightarrow{i/o} s'$. If we consider a probabilistic extension of finite state machines, a transition $s \xrightarrow{i/o/p} s'$ indicates that the probability with which the event happens is p. We consider a variant of the *reactive* interpretation of probabilities (see for example [24]) since it is the most suitable for our framework. Intuitively, a reactive interpretation imposes a probabilistic relation among transitions labelled by the same action but choices between different actions are not quantified. In our setting we are able to express probabilistic relations between transitions outgoing from a state and having the same input action (the output may vary). For example, let us suppose that the transitions from state s are $t_1 = s \xrightarrow{i_1/o_1/p_1} s_1$, $t_2 = s \xrightarrow{i_1/o_2/p_2} s_2$, $t_3 = s \xrightarrow{i_1/o_3/p_3} s_2$, $t_4 = s \xrightarrow{i_2/o_1/p_4} s_3$, and $t_5 = s \xrightarrow{i_2/o_3/p_5} s_1$. If input i_1 is received then the choice between t_1, t_2, and t_3 will be resolved according to probabilities p_1, p_2, and p_3. Naturally, these values must lie between 0 and 1 and their sum should be 1. Something similar happens for t_4 and t_5. However, there does not exist any probabilistic relation between transitions labelled with different input actions (e.g. t_1 and t_4).

After describing a formalism to deal with these concepts we present a *testing methodology* based on mutation testing. Originally *mutation testing* was applied to code [20, 5] but some work has looked at *specification mutation* [6]. Here the specification is mutated and for each *mutant* a test is derived that distinguishes the behaviours of the mutated and original specifications. The effect is to ensure that the implementation under test (IUT) does not implement any of the incorrect specifications. Mutations are chosen in order to simulate real faults. The belief is that if a test suite distinguishes between the specification and mutants then it distinguishes between the specification and any faulty IUT. We

*This research was partially supported by the Spanish MEC project WEST/FAST TIN2006-15578-C02 and the Marie Curie RTN *TAROT* (MRTN-CT-2003-505121).

0-7695-2984-4/07 $25.00 © 2007 IEEE
DOI 10.1109/TAIC.PART.2007.20

describe different *mutation operators* that can be applied to a PFSM specification. Additionally, we present approaches to finding input sequences in order to distinguish the mutants and the specification.

This paper concerns black-box testing; if we apply an input to an IUT then we observe an output but we cannot *see* the probabilities that the IUT has assigned to the choices. Thus, even though implementations will behave according to fixed probabilities we cannot determine their values through testing. In our approach, we *estimate* the probabilities by applying a test several times. We use *statistical results* to establish the number of times we need to apply the test to obtaining a required confidence level.

The rest of the paper is organized as follows. In the next section we introduce preliminary concepts and the notion of a PFSM. In Section 3 we show how we can produce input sequences that distinguish states of a PFSM. In Section 4 we introduce mutation operators for PFSMs and corresponding test generation methods. In Section 5 we describe how testing can use input sequences produced by the methods in Section 4. In Section 6 we review previous works on testing probabilistic systems. Finally, in Section 7 we present our conclusions and some lines of future work.

2 Preliminaries

2.1 Basic notation

In this paper sequences are represented by listing their elements preceded by $\langle$, followed by $\rangle$, and separated by commas. Where a variable represents a sequence its name will have a bar above it, an example being $\bar{a}$. In addition, $[0, 1]$ denotes the set $\{p \mid 0 \leq p \leq 1\}$ of numbers that could represent probabilities, $(0, 1]$ denotes the set $\{p \mid 0 < p \leq 1\}$ of numbers that could represent positive probabilities, and $(0, 1)$ denotes the set of values strictly between 0 and 1 and so $(0, 1) = \{p \mid 0 < p < 1\}$.

Given set X, $\mathcal{P}(X)$ denotes the powerset of X: the set of subsets of X. Thus, $\mathcal{P}(X) = \{X' | X' \subseteq X\}$. Given set W of sequences, $Pre(W) = \{\bar{x}' | \exists \bar{x} \in W, \bar{x}'' \in X^*.\bar{x} = \bar{x}'\bar{x}''\}$ denotes the set of prefixes of sequences from W. Given sets A and B, $A \leftrightarrow B$ denotes the set of relations between A and B. Given a relation f of type $A \leftrightarrow B$ and $a \in A$, a is related to b under f is denoted by $f(a, b)$ and $f(a)$ denotes the set of elements of B related to a under f and so $f(a) = \{b \in B | f(a, b)\}$.

2.2 Mutation testing

The idea behind mutation testing is that if a test suite distinguishes a program P from other similar programs then it is probably good at discovering faults. The technique introduces small changes in a program, one at a time, to generate a set of *mutants*. We produce mutants by applying one or more *mutation operators* to a given program. In general, P' is an *nth order mutant* if it is produced by application a sequence of n mutation operators. Usually only *first order mutants* are considered and two arguments are used to justify this. First, the *competent programmer hypothesis* states that expert programmers often write almost correct programs, so low order mutants represent most real faults. Second, if the tests find small differences generated by low order mutants, then it is likely that they find more complex differences. This is called the *coupling effect*.

After we have obtained a collection of mutants from a program, a set of tests T is applied to distinguish each of the mutants from the original program. If the output produced by a mutant P' is different to the one produced by the original program P for test $t \in T$, then t *kills* P'. If no possible test case kills P', then P' is an *equivalent mutant* of P. The objective of mutation analysis is to produce test cases that kill all non-equivalent mutants. Test suites that achieve this goal are *adequate relative to mutation*.

Another strategy given in [6] is *specification mutation* . The specification is mutated, and for each mutation a test is derived that distinguishes the behaviours of the mutated and original specifications. The effect is to ensure that the system under test does not implement any of the incorrect specifications. The mutations are chosen in order to simulate real faults and thus the belief is that a test suite that kills the mutants will not be passed by a faulty system. This approach has also been applied with finite state machines based models [12, 11, 13, 33]).

2.3 Finite Automata

A Finite Automaton (FA) N is defined by a tuple (S, s_0, A, δ, S_F) in which S is a finite set of states, $s_0 \in S$ is the initial state, A is the finite alphabet, $\delta : S \times A \leftrightarrow S$ is the state transfer relation, and $S_F \subseteq S$ is the set of final states. If N receives $a \in A$ when in state $s \in S$ it moves to a state $s' \in \delta(s, a)$ and this defines a transition (s, s', a). The relation δ can be extended to take sequences from A^*, giving relation δ^*, in the usual way. FA N is a deterministic finite automaton (DFA) if for all $a \in A$ and $s \in S$, $|\delta(s, a)| \leq 1$.

State s of N defines the language $L_N(s) = \{\bar{a} \in A^* | \delta^*(s, \bar{a}) \cap S_F \neq \emptyset\}$ of words that can take N from s to a final state. Word $\bar{a} \in A^*$ *distinguishes* states s and s' of N if $\bar{a}$ is in exactly one of $L_N(s)$ and $L_N(s')$. If no word distinguishes s and s' then they are *equivalent*. Two FA are *equivalent* if their initial states are equivalent. DFA N is *minimal* if no DFA with fewer states is equivalent to N.

A Probabilistic Finite Automaton (PFA) N is defined by a tuple $(S, s_0, A, \delta, S_F, prob)$ in which S is a finite set of states, $s_0 \in S$ is the initial state, A is the finite alphabet,

$\delta : S \times A \leftrightarrow S$ is the state transfer relation, $S_F \subseteq S$ is the set of final states, and $prob$ is the transition probability function of type $S \times A \times S \to [0,1]$. If N receives $a \in A$ when in state $s \in S$ it moves to a state $s' \in \delta(s,a)$ with probability $prob(s,a,s')$ and this defines transition (s,s',a). The relation δ can be extended to take sequences from A^*, giving δ^*, in the usual way.

Definition 1 *Let $N = (S, s_0, A, \delta, S_F, prob)$ be a PFA and let s, s' be states of N. Then $\bar{a} \in A^*$* distinguishes *states s and s' if the string $\bar{a}$ is accepted from states s and s' with different probabilities. States s and s' are* equivalent *if no string from A^* distinguishes them.*

2.4 Probabilistic finite state machines

A non-deterministic finite state machine (NFSM) is a FA in which each transition has an associated output. An NFSM is defined by a tuple (S, s_0, X, Y, f) in which S is a finite set of states, $s_0 \in S$ is the initial state, X is the finite input alphabet, Y is the finite output alphabet, and f is the transition relation. For each state $s \in S$ and input $x \in X$, $f(s,x)$ denotes a set of tuples of the form (s', y) in which $s' \in S$ and $y \in Y$. Given $(s', y) \in f(s,x)$, $(s, s', x/y)$ is a *transition* and this should be interpreted as meaning that if we receive input x while in state s then we can move to state s' and produce output y. A deterministic finite state machine (DFSM) is an NFSM in which for every state s and input x, $|f(s,x)| \leq 1$. There has been much interest in testing from a DFSM (see, for example, [9, 18, 3]) or an NFSM (see, for example, [26, 29, 21, 17]). See [25] for a survey.

A probabilistic finite state machine (PFSM) is an NFSM in which every transition also has an associated probability. A PFSM M is defined by a tuple (S, s_0, X, Y, h) in which S is a finite set of states, $s_0 \in S$ is the initial state, X is the finite input alphabet, Y is the finite output alphabet, and $h : S \times X \leftrightarrow S \times Y \times (0,1]$ is the transition relation. For each state $s \in S$ and input $x \in X$, $h(s,x)$ denotes a set of tuples of the form (s', y, p) in which $s' \in S$, $y \in Y$, and $p \in (0,1]$[1]. For all $s \in S$ and $x \in X$, $\sum_{(s',y,p) \in h(s,x)} p = 1$.

If $(s', y, p) \in h(s,x)$ then (s, x, y, s', p) is a transition of M with starting state s. The probabilities should be interpreted in the following way. If $(s', y, p) \in h(s,x)$ and M receives input x when in state s then with probability p it moves to state s' and produces output y. For a survey on probabilistic automata see [36].

[1] An equivalent but less compact representation would include in $h(s,x)$ the transitions with probability 0.

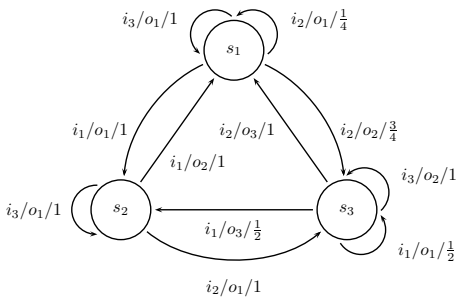

Figure 1. A PFSM

Example 1 *Let us consider the probabilistic finite state machine depicted in Figure 1. Each transition has an associated probability. We can observe that all the transitions from state s_2 have probability* 1. *In contrast, for the state s_1 we have the transitions $(s_1, i_2, o_1, s_2, \frac{1}{4})$ and $(s_1, i_2, o_2, s_3, \frac{3}{4})$ with input i_2; naturally their probabilities sum to* 1.

An alternative characterization of the transitions of M is through a function p_M of type $S \times X \times Y \times S \to [0,1]$. Given states s and s', input s and output y, $p_M(s,x,y,s')$ is the probability that M moves to state s' and produces output y if it receives input x when in state s. Naturally h and p_M fully define one another since $(s', y, p) \in h(s,x) \Leftrightarrow p > 0 \wedge p_M(s,x,y,s') = p$.

We can extend the transition relation h to input sequences, producing relation h^* of type $S \times X^* \leftrightarrow S \times Y^* \times (0,1]$. It is simplest to first define p^*_M and express h^* in terms of p^*_M. A similar approach has been applied for PFA (see, for example, [36]).

Definition 2 *Let $M = (S, s_0, X, Y, h)$ be a PFSM. Given input/output sequence $\bar{x}/\bar{y}$ and state $s \in S$ we define the probability of reaching state s' from s with $\bar{x}/\bar{y}$ as:*

$$p^*_M(s, \epsilon, y, s') = \begin{cases} 1 & \text{if } y = \epsilon \text{ and } s' = s, \\ 0 & \text{otherwise} \end{cases}$$

$$p^*_M(s, \bar{x}x, \bar{y}y, s') = \sum_{s'' \in S} p^*_M(s, \bar{x}, \bar{y}, s'') p_M(s'', x, y, s')$$

Let us note that $p_M(s, \bar{x}, \bar{y}, s')$ is 0 whenever $|\bar{x}| \neq |\bar{y}|$. In a slight abuse of notation $p^*_M(s, \bar{x}, \bar{y})$ denotes the probability that M produces output sequence $\bar{y}$ if it receives input sequence $\bar{x}$ when in state s. Thus, $p^*_M(s, \bar{x}, \bar{y}) = \sum_{s' \in S} p^*_M(s, \bar{x}, \bar{y}, s')$. We can now extend the transition relation h to input sequences.

$$h^*(s, \bar{x}) = \{(s', \bar{y}, p) | p = p^*_M(s, \bar{x}, \bar{y}, s') \wedge p > 0\}$$

PFSM M is *observable*, or *output-complete*, if for every state s, input x and output y there is at most one transition leaving s with input x and output y. In this paper we consider both observable PFSMs (OPFSMs) and PFSMs that are not observable. PFSM M is *completely specified* if for every state s and input x, $|h(s,x)| \geq 1$ holds.

Some systems have a special operation called a reset that takes the system to its initial state irrespective of the current state. The IUT has a *reliable reset* if it has a reset that is known to be correct. A reliable reset could represent some way of resetting the IUT, such as switching it off and then on again. In this paper we assume that the IUT has a reliable reset.

Two states of an NFSM or DFSM are equivalent if they define the same sets of behaviours: the same set of input/output sequences. For two states of an PFSM to be equivalent we need that they define the same sets of behaviours with the same probabilities. Thus, states s and s' of PFSM M are *equivalent* if for every input sequence $\bar{x} \in X^*$ and output sequence $\bar{y} \in Y^*$, we have that $p^*_M(s,\bar{x},\bar{y}) = p^*_M(s',\bar{x},\bar{y})$. If s and s' are not equivalent then they are *distinguishable*. Two PFSMs are equivalent if their initial states are equivalent. If there exists some $\bar{y} \in Y^*$ such that $p^*_M(s,\bar{x},\bar{y}) \neq p^*_M(s',\bar{x},\bar{y})$ then $\bar{x}$ is said to *distinguish* s and s'. M is minimal if no PFSM with fewer states that M is equivalent to M.

Let us suppose that we are testing a black box that is equivalent to M and we know that we are either in state s or in state s'. If $\bar{x}$ distinguishes s and s' then it can be used to determine the state if we can apply it in the current state multiple times since we can estimate the probability of each output sequence and the corresponding probabilities are different for s and s'. If we have a reset then we could repeat the following separated by resets: apply the test sequence that led to the current state and then apply $\bar{x}$. If we are to apply an input sequence $\bar{x}$ once only to distinguish two states s and s' then we need a stronger concept: the input of $\bar{x}$ must be guaranteed to lead to different output sequences from s and s' and thus the corresponding sets of possible output sequences must be disjoint.

When reasoning about testing it is normal to assume that the IUT behaves like an unknown element of a fault model (see, for example [22]). Usually the fault model contains descriptions written in the same language as the specification. Thus, for example, when testing from a DFSM it is normal to assume that the IUT behaves like an unknown DFSM. Conformance relations can then be formally defined. Here we briefly review the conformance relations for testing from a (completely specified) DFSM or NFSM.

When testing from a completely specified NFSM there are two notions of correctness. One notion is equivalence, but an alternative is that every behaviour of the IUT is also a behaviour of the specification and that the IUT is completely specified. The assignment of probabilities to transitions removes the possibility of using the second notion of correctness for NFSMs and thus when testing from a PFSM the IUT is correct if it behaves like an unknown PFSM N that is equivalent[2] to M.

3 Distinguishing states of a PFSM

This section shows how we can produce an input sequence that distinguishes two states of PFSM M. We first consider the case where M is observable and show that here the problem can be represented in terms of finding a sequence that distinguishes two states of a FA. We then consider the general case.

Definition 3 *Let $M = (S, s_0, X, Y, h)$ be a PFSM. We define a FA $F(M) = (S, s_0, A, \delta, S)$ where A is $X \times Y \times (0,1]$ and given $s \in S$, $x \in X$, $y \in Y$, and $p \in (0,1]$, $\delta(s,(x,y,p)) = s'$ if and only if $(s',y,p) \in h(s,x)$.*

Given $\bar{a} \in A^*$, let $in(\bar{a})$ denote the corresponding input sequence. The function in can be defined recursively in the following way. First, the base case is $in(\epsilon) = \epsilon$. Given $a = (x,y,p) \in A$ and $\bar{a} \in A^*$, $in(a\bar{a}) = xin(\bar{a})$. The following show that algorithms that produce sequences that distinguish states of a FA can also be used to produce sequences that distinguish state of an OPFSM.

Proposition 1 *If input sequence $\bar{x}$ distinguishes states s and s' of OPFSM M then there is some $\bar{a} \in A^*$ such that $\bar{a}$ distinguishes states s and s' of $F(M)$ and $in(\bar{a}) = \bar{x}$.*

Proof

Let us suppose that $\bar{x}$ distinguishes states s and s' of M. By definition, there exists $\bar{y} \in Y^*$ such that $p^*_M(s,\bar{x},\bar{y}) \neq p^*_M(s',\bar{x},\bar{y})$. Without loss of generality, assume that $p = p^*_M(s,\bar{x},\bar{y}) > 0$.

Since M is observable, there is only one walk in M from state s with input/output sequence $\bar{x}/\bar{y}$. Let $\langle t_1,\ldots,t_k\rangle$ ($k = |\bar{x}|$) denote the sequence of transitions on this walk and for each $t_i = (s_i,x_i,y_i,s_{i+1},p_i)$ let $a_i = (x_i,y_i,p_i)$ and let $\bar{a} = \langle a_1,\ldots,a_k\rangle$. If there is no walk from s' in M with input/output sequence $\bar{x}/\bar{y}$ then $\bar{a}$ does not label a walk from state s' of $F(M)$ and so $\bar{a}$ distinguishes states s and s' of $F(M)$ as required. Alternatively, if there is a walk from s' in M with input/output sequence $\bar{x}/\bar{y}$ then there is only one such walk so since $p^*_M(s,\bar{x},\bar{y}) \neq p^*_M(s',\bar{x},\bar{y})$, $\bar{a}$ does not label a walk from state s' of $F(M)$ and so $\bar{a}$ distinguishes states s and s' of $F(M)$ as required. □

Proposition 2 *Given an OPFSM M and $\bar{a} \in A^*$ such that $\bar{a}$ distinguishes states s and s' of $F(M)$, if no proper prefix*

[2] A richer set of conformance relations have been defined for general probabilistic state machines (see, for example, [31, 37]).

of $\bar{a}$ distinguishes s and s' and $in(\bar{a}) = \bar{x}$ then $\bar{x}$ distinguishes states s and s' of M.

Proof

Let us suppose that $\bar{a}$ distinguishes states s and s' of $F(M)$ and no proper prefix of $\bar{a}$ distinguishes s and s'. Let $\bar{a} = \langle a_1, \ldots, a_k \rangle$, $a_i = (x_i, y_i, p_i)$, $\bar{x} = \langle x_1, \ldots x_k \rangle$ and $\bar{y} = \langle y_1, \ldots, y_k \rangle$. Without loss of generality, $\bar{a}$ labels a walk from state s of $F(M)$ and does not label a walk from state s' of $F(M)$.

Since M is observable, there is only one walk in M from state s with input/output sequence $\bar{x}/\bar{y}$. If $\bar{x}/\bar{y}$ does not label a walk from state s' of M then by definition $\bar{x}$ distinguishes states s and s' of M as required. We thus assume that $\bar{x}/\bar{y}$ labels a walk from state s' of M and that this walk consists of the sequence $\langle t'_1, \ldots, t'_k \rangle$ of transitions. Let p'_i denote the probability associated with transition t'_i. Since no proper prefix of $\bar{a}$ distinguishes s and s' and so $\langle a_1, \ldots, a_{k-1} \rangle$ labels a walk from s' in $F(M)$ we have that $p_i = p'_i$ for $1 \leq i < k$. Since $\bar{a}$ does not label a walk from s' in $F(M)$, $p_k \neq p'_k$. Thus, $p^*_M(s, , \bar{x}, \bar{y}) = p_1 \ldots p_k \neq p'_1 \ldots p'_k = p^*_M(s', \bar{x}, \bar{y})$ and so $\bar{x}$ distinguishes states s and s' of M as required. □

Note the condition that no proper prefix of $\bar{a}$ distinguishes s and s'. To see why we require this suppose that $\bar{a} = (x_1, y_1, 0.5)(x_2, y_2, 0.5)$, $\bar{a}$ labels a walk from s, and there is a walk with label $(x_1, y_1, 1)(x_2, y_2, 0.25)$ from state s'. Then $\bar{a}$ distinguishes states s and s' of $F(M)$, since $\bar{a}$ labels a walk from s but not s'. However, the input sequence x_1x_2 might not distinguish between states s and s' of M since both states have a probability of 0.25 of producing y_1y_2. Naturally, if we consider the minimal prefix of $\bar{a}$ that distinguishes states s and s' of $F(M)$ then the corresponding input sequence x_1 does distinguish between states s and s' of M.

Proposition 3 *There is an algorithm running in time $O(n^2)$ that takes two states s_1 and s_2 of OPFSM M and determines whether they are equivalent, where n is the number of states of M. If s_1 and s_2 are distinguishable then the algorithm returns a minimal input/output sequence of length no more than $n - 1$ that distinguishes them.*

Proof

Since $F(M)$ is a minimal FA, there exists a set of sequences of length at most $n - 1$ that pairwise distinguish the states of $F(M)$ and such a set can be found in $O(n^2)$ time (see, for example, [15]). The result thus follows from Proposition 2. □

More general results have been proved for probabilistic FA that are not deterministic and thus for PFSM that are not observable[3]. The following has been proved [34].

[3] A PFSM is observable if and only if the corresponding FA is deterministic.

Theorem 1 *There is an algorithm running in time $O((n_1 + n_2)^4)$ that takes two probabilistic automata U_1 and U_2 and determines whether U_1 and U_2 are equivalent, where n_1 and n_2 are the number of states of U_1 and U_2 respectively. Furthermore, if U_1 and U_2 are not equivalent then the algorithm outputs the lexicographically minimum string that is accepted by U_1 and U_2 with different probabilities. This string will always be of length at most $n_1 + n_2 - 1$.*

We now show how this result can be applied to PFSMs.

Definition 4 *Given PFSM $M = (S, s_0, X, Y, h)$ we can define a PFA $F_P(M) = (S, s_0, A, \delta, S, prob)$ in which*

1. *The alphabet A is $X \times Y$.*
2. *Given $s \in S$, $x \in X$, $y \in Y$, and $p \in (0, 1]$, $\delta(s, (x, y)) = s'$ and $prob(s, (x, y), s') = p$ if and only if $(s', y, p) \in h(s, x)$.*

The following results are an immediate consequence of Definitions 1 and 4.

Proposition 4 *If input sequence $\bar{x}$ distinguishes two states s and s' of PFSM M then there is some $\bar{a} \in A^*$ such that $\bar{a}$ distinguishes states s and s' of PFA $F_P(M)$ and $in(\bar{a}) = \bar{x}$.*

Proposition 5 *Given PFSM M and input sequence $\bar{x}$, if there is some $\bar{a} \in A^*$ such that $\bar{a}$ distinguishes states s and s' of $F_P(M)$, no proper prefix of $\bar{a}$ distinguishes s and s', and $in(\bar{a}) = \bar{x}$ then $\bar{x}$ distinguishes two states s and s' of M.*

Proposition 6 *There is an algorithm running in time $O((n_1 + n_2)^4)$ that takes as input two PFSMs M_1 and M_2 and determines whether M_1 and M_2 are equivalent, where n_1 and n_2 are the number of states of M_1 and M_2 respectively. If M_1 and M_2 are not equivalent then the algorithm outputs the lexiographically minimum input/output sequence for which M_1 and M_2 have different probabilities. This input/output sequence will always be of length at most $n_1 + n_2 - 1$.*

Proof

This follows by applying Theorem 1 to $F_P(M)$ and $F_P(M')$. □

Since our PFSM M is minimal, and so its states are pairwise distinguishable, we can define a set of input sequences that distinguish between the states of M.

Definition 5 *A set W of input sequences is a characterization set for PFSM M if for every pair (s, s') of states of M with $s \neq s'$ there exists some input sequence $\bar{x} \in W$ that distinguishes s and s'.*

The proof of the following is similar to that for the equivalent result for DFSMs (see, for example, [9]).

Proposition 7 *Let us suppose that M is a minimal OPFSM with n states. Then there exists a characterization set W for M with at most $n-1$ input sequences where each sequence has length at most $n-1$.*

The more general case, where M need not be observable, is similar.

Proposition 8 *Let us suppose that M is a minimal PFSM with n states. Then there exists a characterization set W for M with at most $n-1$ input sequences where each sequence has length at most $2n-1$.*

Proof

States s and s' of M are distinguished by an input sequence $\bar{x}$ if and only if $\bar{x}$ distinguishes between the PFSM formed by changing the initial state of M to s and the PFSM formed by changing the initial state of M to s'. Thus, by Proposition 6, there is a sequence of length at most $2n-1$ that distinguishes between any pair of distinct states of M.

Let $W = \{\bar{x}_1, \ldots \bar{x}_k\}$ denote a characterization set for M such that no proper subset of W is a characterization set for M. Let $\sim_i$ $(0 \leq i \leq k)$ denote the equivalence relation on the states of M such that: $s \sim_i s'$ if for all $1 \leq j \leq i$ we have that $\bar{x}_j$ does not distinguish between s and s'. Clearly $\sim_1$ has at least two equivalence classes and by the minimality of W we must have that for all $1 \leq i < k$, $\sim_{i+1}$ has more equivalence classes than $\sim_i$. Thus, the number of equivalence classes for $\sim_i$ $(1 \leq i \leq k)$ must be at least $i+1$. However, the number of equivalence classes is bounded above by n and thus $k+1 \leq n$ and so $k \leq n-1$ as required. □

Similar to testing from a DFSM [14], when identifying a given state s_i of M it may be sufficient to use a set of prefixes of sequences in W. Such a set is called an identification set.

Definition 6 *Given state s_i of minimal PFSM M and characterization set W, a set $W_i \subseteq Pre(W)$ is an* identification set *if for every state s_j of M with $s_i \neq s_j$, there is some input sequence $\bar{x} \in W_i$ that distinguishes s_i and s_j.*

4 Mutation operators

This section describes mutation operators for PFSMs and approaches to finding input sequences to distinguish the resultant mutants. Section 5 explains how testing can proceed on the basis of this. Throughout this section $M = (S, s_0, X, Y, h)$ denotes the PFSM being mutated and M' denotes a mutant. Proposition 6 tell us that we can decide whether M and M' are equivalent in time $O(n^4)$ and, if they are not equivalent, produce a sequence of length at most $2n-1$ to distinguish them. In this section we consider conditions under which we can improve on this.

4.1 Changing the initial state

We can form a mutant of M by making some state $s \in S \setminus \{s_0\}$ of M the initial state and this gives $n-1$ different mutants. Let $M_s = (S, s, X, Y, h)$ $(s \neq s_0)$ be such a mutant. Then we want to find a sequence that distinguishes the initial states of M and M_s. This is equivalent to the problem of finding a sequence to distinguish states s_0 and s of M. The following result is thus clear.

Proposition 9 *Let $M = (S, s_0, X, Y, h)$ be a PFSM and let $M_s = (S, s, X, Y, h)$ for some state $s \neq s_0$. If W_0 is an identification set for the initial state of M then there exists an input sequence $\bar{x} \in W_0$ that distinguishes M and M_s.*

Thus, any identification set W_0 for the initial state of M distinguishes M from every mutant of the form M_s for $s \neq s_0$. As shown in Section 3, $|W_0| \leq n-1$ and if M is observable then the elements of W_0 have length at most $n-1$ and otherwise they have length at most $2n-1$.

4.2 Altering probabilities

Suppose that $t = (s, x, y, s', p)$ is a transition of M and let Δ be a (possibly negative) value such that $0 \leq p + \Delta \leq 1$. We can mutate M by changing the probability associated with t to $p + \Delta$. Naturally, we must change the probability of at least one other transition from s with input x so that the sum of the probabilities is still 1. Let M' be a PFSM formed by altering the probability of t to $p' \neq p$.

Let us suppose that the probability of producing output y from state s of M in response to x is different from the probability of producing output y from state s of M' in response to x (this must be the case if M is observable). Then to distinguish M and M' it is sufficient to devise a sequence $\bar{a}$ in the following way. First, find a shortest path $\bar{a}_1$ in $F(M)$ from s_0 to s. Then we set $\bar{a} = \bar{a}_1(x, y, p)$.

Proposition 10 *Let $M(t, \Delta)$ be an PFSM formed by altering the probability of transition $t = (s, x, y, s', p)$ of PFSM M to $0 \leq p + \Delta \leq 1$ and suppose that the probability of producing output y from state s of M in response to x is different from the probability of producing output y from state s of M' in response to x (i.e. $p_M(s, x, y) \neq p_{M(t,\Delta)}(s, x, y)$). Let $\bar{a}_1$ be a shortest path in $F(M)$ from s_0 to s. If $\bar{a} = \bar{a}_1(x, y, p)$ then $in(\bar{a})$ distinguishes M and M' and has length at most n.*

Proof

Let $\bar{x}_1$ and $\bar{y}_1$ be the input and output sequences from $\bar{a}_1$ respectively. By the minimality of $\bar{a}_1$, no path in $F(M)$ from s_0 with label $\bar{a}_1$ contains the transition t. Thus, $p^*_M(s_0, \bar{x}_1, \bar{y}_1) = p^*_{M(t,\Delta)}(s_0, \bar{x}_1, \bar{y}_1)$. Thus, since $p_M(s, x, y) \neq p_{M(t,\Delta)}(s, x, y)$ and for every state $s' \neq s$ we have that $p_M(s', x, y) = p_{M(t,\Delta)}(s', x, y)$ we have that $p^*_M(s_0, \bar{x}x, \bar{y}y) \neq p^*_{M(t,\Delta)}(s_0, \bar{x}_1x, \bar{y}_1y)$ as required. Finally, since $\bar{a}_1$ is a shortest path from s_0 to s it has length at most $n-1$ (since there are no repeated states) and so $\bar{a}$ has length at most n. □

If the probabilities of producing output y in response to x from state s of M and $M(t, \Delta)$ are the same then by Proposition 6 we can decide in $O(n^4)$ whether M and $M(t, \Delta)$ are equivalent and, if they are not, find a sequence of length at most $2n-1$ that distinguishes them.

4.3 Changing the target state of a transition

Suppose $t = (s, x, y, s', p)$ is a transition of M and let s'' denote a state of M ($s' \neq s''$). We can create a new PFSM, called $M(t, s'')$, by changing the ending state of t to s''. The following are clear.

Proposition 11 *Let t denote a transition (s, x, y, s', p) of PFSM M and let s'' be a state of M with $s' \neq s''$. If M is observable then $M(t, s'')$ is observable.*

Proposition 12 *If M is observable then there is an $O(n^2)$ time algorithm that decides whether $M(t, s'')$ and M are equivalent and, if they are not equivalent, returns an input sequence of length at most $2n-1$ that distinguishes them.*

Naturally, if M is not observable and M and $M(t, s'')$ are not equivalent then in $O(n^4)$ we can produce an input sequence $\bar{x}$, of length at most $2n-1$, that distinguishes them.

4.4 Creating a new transition

Let us suppose that $t = (s, x, y, s', p)$ is a transition of M, let s'' denote a state of M, let y' denote an output and let $p' < p$ denote a probability. We can create a new PFSM, $M(t, s'', y', p')$, by reducing the probability associated with t to $p - p'$ and creating a new transition (s, x, y', s'', p'). If M has a transition from s to s'' with input x and output y' then we have simply altered probabilities and simulated a mutation operator already discussed in Subsection 4.2. Since this case is redundant we do not consider it here.

If $y' \neq y$ then we have reduced the probability of producing output y in response to x from s. Thus, we can distinguish M and $M(t, s'', y', p')$ by choosing a minimum length sequence $\bar{a}_1 \in A^*$ that labels a path in $F(M)$ from s_0 to s and use $in(\bar{a}_1)x$. This input sequence has length at most n. Otherwise we can refer to the result that we can decide in $O(n^4)$ time whether M and $M(t, s'', y', p')$ are equivalent and, if they are not, produce an input sequence of length at most $2n-1$ that distinguishes them.

Example 2 *Figure 2 shows two mutants of the PFSM from Figure 1. The first is formed by changing the probability associated with transition $(s_3, i_1, o_1, s_3, \frac{1}{2})$ to $\frac{1}{4}$. Since the sum of probabilities of the transitions from s_3 with input i_1 must be 1, we also alter the probability attached to transition $(s_3, i_1, o_3, s_2, \frac{1}{2})$ to $\frac{3}{4}$. The second mutant is obtained from by adding a new transition $(s_2, i_3, o_3, s_3, \frac{2}{5})$. This forces us to decrease the probability associated to the transition $(s_2, i_3, o_1, s_2, 1)$ to $\frac{3}{5}$.*

5 Applying the test sequences

Let M denote the specification PFSM, M' a mutant of M, and $\bar{x}/\bar{y}$ an input/output sequence such that $p^*_M(s_0, \bar{x}, \bar{y}) \neq p^*_{M'}(s'_0, \bar{x}, \bar{y})$ and so $\bar{x}$ distinguishes between M and M'. Let p_s denote $p^*_M(s_0, \bar{x}, \bar{y})$ and let p_m denote $p^*_{M'}(s'_0, \bar{x}, \bar{y})$. If we can determine the probability p of observing $\bar{y}$ in response to $\bar{x}$ in the IUT then we have two cases: if $p = p_s$ then the IUT is distinguished from the mutant M' and otherwise the IUT is faulty. However, we cannot determine p through testing; the best we can do is to produce an estimate $\hat{p}$ of p.

If we test the IUT with $\bar{x}$ a total of r times and in k of these tests we observe $\bar{y}$ then our estimate is $\hat{p} = \frac{k}{r}$. Naturally, the greater the value of r the higher our confidence in $\hat{p}$ being close to the true value p. We now show how statistical results regarding confidence intervals can be used in order to determine the required value of r.

Suppose that we fix a confidence level $c \in (0, 1)$ and we have this confidence of $\hat{p}$ being within e of p. The confidence denotes the probability that our estimate $\hat{p}$ satisfies $p - e < \hat{p} < p + e$. We want the estimate to either provide evidence that the IUT is faulty or that the IUT is not equivalent to M'. We are guaranteed to achieve this if we cannot have both p_s and p_m in the interval $(\hat{p} - e, \hat{p} + e)$. This is the case if $2e \leq |p_s - p_m|$ and thus we can set $e = \frac{|p_s - p_m|}{2}$. Naturally, we can choose a smaller value of e if we wish to have an estimate $\hat{p}$ with a smaller confidence interval.

Each application of $\bar{x}$ has two possible results: either the output sequence is $\bar{y}$ or it is some $\bar{y}' \neq \bar{y}$. We thus have a binomial distribution. We now discuss two sets of standard statistical results for binomial distributions that show how we can choose r given e and c.

Note that an alternative approach is to use hypothesis testing, with the null hypothesis either being that the true probability is the same as the probability in the mutant ($p = p_m$) or that the true probability is either p_m or 'on

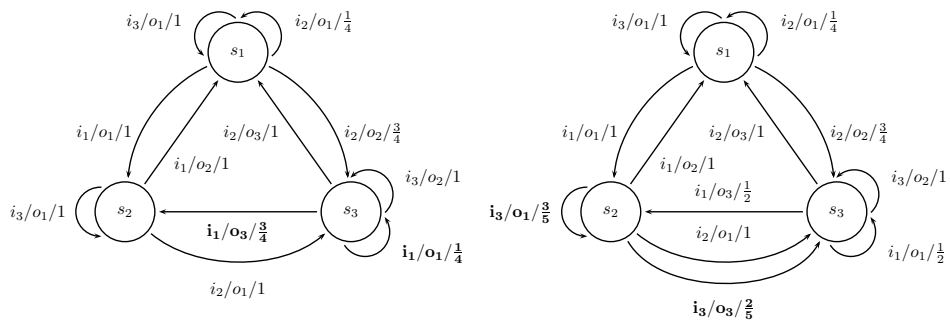

Figure 2. Two mutants

the other side of p_m from p_s' (i.e. if $p_m > p_s$ then the null hypothesis is $p - p_s \geq p_m - p_s$). However, here we focus on the use of confidence intervals since they have the additional benefit of allowing us to state the confidence we have of the true probability p being within e of the sample probability $\hat{p}$.

5.1 Estimating with a small sample

Since we have a binomial distribution with probability p, the probability of observing $\bar{y}$ in response to $\bar{x}$ a total of k times in r trials is

$$P(k, p, r) = \left(\begin{array}{c} r \\ k \end{array} \right) p^k (1-p)^{(r-k)}$$

where

$$\left(\begin{array}{c} r \\ k \end{array} \right) = \frac{r!}{k!(r-k)!}$$

Based on this we get the following likelihood function $L(p', k, r)$, which represents the probability that p' is the true probability if we have observed $\bar{y}$ a total of k times out of r tests with $\bar{x}$.

$$L(p', k, r) = \left(\begin{array}{c} r \\ k \end{array} \right) p'^k (1-p')^{(r-k)}$$

Given k and r it is possible to calculate a value e such that we have confidence of at least c that $\hat{p}$ is within e of p and thus to determine whether we have tested a sufficient number of times. However, this computation becomes increasingly expensive as r increases and this motivates an interest in alternative approaches for large samples.

5.2 Estimating with a large sample

We can treat the mean $\bar{p}$ as a normally distributed random variable if $rp > 5$ and $r(1-p) > 5$ and this distribution has mean p and the following standard deviation (see, for example, [19]).

$$\sqrt{\frac{p(1-p)}{r}}$$

While we do not know p, we can check that $r\hat{p} > 5$ and $r(1-\hat{p}) > 5$ once the estimate has been produced. Here we consider this case.

Given a normal distribution with standard deviation σ, true mean μ and confidence value c there is a value z such that the proportion of the distribution that is within the region $(\mu - z\sigma, \mu + z\sigma)$ is c. For a confidence of 0.95 we can choose the value $z_0 = 1.96$ (see, for example, [19]). Clearly, we should choose r such that $z_0\sigma \leq e$. Since $\sigma = \sqrt{\frac{p(1-p)}{r}}$ we choose r such that

$$\sqrt{\frac{p(1-p)}{r}} z_0 \leq e$$

This can be rewritten to:

$$r \geq \frac{z_0^2 p(1-p)}{e^2}$$

Thus, we apply $\bar{x}$ a total of r times for some r that satisfies the above equation. While we do not know p in advance, we know that the worst case is with $p = \frac{1}{2}$ giving:

$$r \geq \frac{z_0^2}{4e^2}$$

If we require a value of $c = 0.95$, z_0 is slightly less than 2 and so it is sufficient to choose r such that:

$$r \geq \frac{1}{e^2}$$

If $\hat{p}$ is within e of the specified value p_s then we have the required confidence that the IUT is not equivalent to mutant M'; otherwise we have confidence that the IUT is faulty. Naturally, at this point we may wish to test the IUT further with $\bar{x}$ to gain an estimate with a narrower associated confidence interval.

6 Related work

This section describes previous work on testing probabilistic systems. Interestingly, there has been relatively little work on this but there has been a considerable amount of work on model checking PFSMs (see for example [2, 35, 10, 23, 32]).

It has been noted that we can consider the testing of a stateless system to be a process of sampling its behaviour. If the test suite used is randomly generated, possibly based on a distribution (operational profile) that reflects expected usage then the result of testing can be used to estimate the reliability of the IUT [8, 4]. Further, we can place a confidence in the observed reliability reflecting the true reliability of the IUT within some margin.

Researchers have tackled the problem of testing from an observable NFSM when each transition introduces a random delay and the expected delay for a transition is represented by a probability distribution [28]. The problem is to test to determine whether the distribution for each transition in the IUT is correct, where correctness is represented by a range of conformance relations. Testing is used to check that the IUT satisfies the conformance relation, relative to the specification, within a given degree of confidence.

It is sometimes desirable to have a process that can be applied once in order to take a conforming IUT to a given state or (strongly) distinguish two states of the specification and thus of a conforming IUT. Naturally, there need not exist single sequences that are guaranteed to achieve this and thus it is normal to apply an adaptive process. Alur et al. [1] show that it is PSPACE-complete to determine whether there is a single input sequence that strongly distinguishes two states of a PFSM and that it is EXPTIME-complete to determine whether there is an adaptive process that strongly distinguishes two states of a PFSM. Zhang and Zheung show how policies (adaptive processes) can be generated to move an OPFSM from state s to another state s' and how a policy can be found to (strongly) distinguish two states of an OPFSM [38]. Note that the work of Alur et al. [1] and Zhang and Zheung [38] refer to producing a single sequence or policy that will achieve the desired results through one application only. In contrast, we assume that there is a reliable reset operation and thus that we can *repeatedly* apply an input sequence or policy.

A related problem is machine identification; we wish to model the IUT rather than test to check that the IUT conforms to a given model. This problem has been considered in the context of probabilistic state machines [30]. In this approach, the set of observed traces is used to induce a FA in the classical way. The probabilities on each transition are then estimated by determining the ratios of the labels (from each state) observed in testing.

7 Conclusions

This paper developed mutation testing techniques for probabilistic finite state machines (PFSMs). It defined several mutation operators and adapted results from the theory of probabilistic finite automata in order to produce test sequences that distinguish a PFSM M from a mutant M'. An important property of PFSMs is that given two PFSMs M and M' we can decide in polynomial time whether M and M' are equivalent and so mutation testing for PFSMs does not suffer from the equivalent mutant problem.

An input sequence $\bar{x}$ kills a mutant M' if there is some output sequence $\bar{y}$ such that the probabilities of observing $\bar{x}/\bar{y}$ in M and M' are different. When $\bar{x}$ is used in testing we observe resultant input/output sequences and ideally we would like to compare the probabilities of each of these with those expected. However, we cannot determine the true probabilities through testing and so this paper has shown how results from statistic sampling theory can be used to estimate the probabilities with sufficient precision, up to a required level of confidence.

We have shown how an input sequence can be efficiently generated to kill a mutant M' of M. However, in applying the resultant input sequences the number of repetitions depends on the probabilities of the corresponding input/output sequence in M and M'. There thus remains the following question: how can we produce an input sequence $\bar{x}$ that kills M' and minimizes the test execution effort for a given required confidence interval c?

References

[1] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *27th ACM Symp. on Theory of Computing, STOC'95*, pages 363–372. ACM Press, 1995.

[2] C. Baier, M. Kwiatkowska, and G. Norman. Computing probability bounds for linear time formulas over concurrent probabilistic systems. In *PROBMIV'98, Electronics Notes in Theoretical Computer Science 22*, 1999.

[3] M. Barnett, W. Grieskamp, L. Nachmanson, W. Schulte, N. Tillmann, and M. Veanes. Towards a tool environment for model-based testing with AsmL. In *3rd Int. Workshop on Formal Approaches to Testing of Software, FATES'03, LNCS 2931*, pages 252–266. Springer, 2003.

[4] G. Bernot, L. Bouaziz, and P. L. Gall. A theory of probabilistic functional testing. In *19th IEEE Int. Conf. on Software Engineering, ICSE'97*, pages 216–226. ACM Press, 1997.

[5] L. Bottaci and E. Mresa. Efficiency of mutation operators and selective mutation strategies: An empirical study. *Software Testing, Verification and Reliability*, 9(4):205–232, 1999.

[6] D. Carrington and P. Stocks. A tale of two paradigms: Formal methods and software testing. In *Z User Workshop*, pages 51–68. Springer, Workshops in Computing, 1994.

[7] D. Cazorla, F. Cuartero, V. Valero, F. Pelayo, and J. Pardo. Algebraic theory of probabilistic and non-deterministic processes. *Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.

[8] T. Y. Chen and Y. T. Yu. On the expected number of failures detected by subdomain testing and random testing. *IEEE Transactions on Software Engineering*, 4(2):109–119, 1996.

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

[10] J.-M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to ltl model checking of probabilistic systems. In *10th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'03, LNCS 2850*, pages 361–375. Springer, 2003.

[11] S. C. P. F. Fabbri, M. E. Delamaro, J. C. Maldonado, and P. C. Masiero. Mutation analysis testing for finite state machines. In *5th IEEE International Symposium on Software Reliability Engineering (ISSRE 1994)*, pages 220–229, 1999.

[12] S. C. P. F. Fabbri, J. C. Maldonado, P. C. Masiero, M. E. Delamaro, and W. E. Wong. Mutation testing applied to validate specifications based on petri nets. In *Proceedings of the IFIP TC6 Eighth International Conference on Formal Description Techniques (FORTE 95)*, pages 329–337, 1995.

[13] S. C. P. F. Fabbri, J. C. Maldonado, T. Sugeta, and P. C. Masiero. Mutation testing applied to validate specifications based on statecharts. In *10th IEEE International Symposium on Software Reliability Engineering (ISSRE 1999)*, pages 210–219, 1999.

[14] S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite-state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, 1991.

[15] A. Gill. *Introduction to The Theory of Finite State Machines*. McGraw–Hill, 1962.

[16] R. v. Glabbeek, S. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.

[17] R. Hierons. Testing from a non-deterministic finite state machine using adaptive state counting. *IEEE Transactions on Computers*, 53(10):1330–1342, 2004.

[18] R. Hierons and H. Ural. Reduced length checking sequences. *IEEE Transactions on Computers*, 51(9):1111–1117, 2002.

[19] P. G. Hoel. *Elementary Statistics*. John Wiley and Sons, second edition, 1966.

[20] W. Howden. Weak mutation testing and completeness of test sets. *IEEE Transactions on Software Engineering*, 8:371–379, 1982.

[21] I. Hwang, T. Kim, S. Hong, and J. Lee. Test selection for a nondeterministic FSM. *Computer Communications*, 24(12):1213–1223, 2001.

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

[23] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal of Software Tools for Technology Transfer*, 6(2):128–142, 2004.

[24] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[25] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[26] G. 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.

[27] M. Núñez. Algebraic theory of probabilistic processes. *Journal of Logic and Algebraic Programming*, 56(1–2):117–177, 2003.

[28] M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *23rd IFIP WG 6.1 Int. Conf. on Formal Methods for Networked and Distributed Systems, FORTE'03, LNCS 2767*, pages 335–350. Springer, 2003.

[29] A. Petrenko, N. Yevtushenko, and G. v. Bochmann. Testing deterministic implementations from their nondeterministic FSM specifications. In *9th IFIP Workshop on Testing of Communicating Systems, IWTCS'96*, pages 125–140. Chapman & Hall, 1996.

[30] I. Rouvellou and G. W. Hart. Inference of a probabilistic finite state machine from its output. *IEEE Transactions on Systems, Manufacturing and Cybernetics*, 25(3):424–437, 1995.

[31] R. Segala. Testing probabilistic automata. In *7th Int. Conf. on Concurrency Theory, CONCUR'96, LNCS 1119*, pages 299–314. Springer, 1996.

[32] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *16th Int. Conf. on Computer Aided Verification, CAV'04, LNCS 3114*, pages 202–215. Springer, 2004.

[33] T. Sugeta, J. C. Maldonado, and W. E. Wong. Mutation testing applied to validate SDL specifications. In *16th IFIP International Conerence on Testing of Communicating Systems (TestCom 2004)*, volume 2978 of *Lecture Notes in Computer Science*, pages 193–208, 2004.

[34] W. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21:216–227, 1992.

[35] M. Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *5th Int. AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems, ARTS'99, LNCS 1601*, pages 265–276. Springer, 1999.

[36] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines — part 1. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025, 2005.

[37] S.-H. Wu, S. Smolka, and E. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176(1-2):1–37, 1997.

[38] F. Zhang and T. Y. Cheung. Optimal transfer trees and distinguishing trees for testing observable nondeterministic finite-state machines. *IEEE Transactions on Software Engineering*, 29(1):1–14, 2003.