14

TEST GENERATION DRIVEN BY USER-DEFINED FAULT MODELS

I. Koufareva†, A. Petrenko‡ and N. Yevtushenko†

† *Tomsk State University*
36 Lenin St
Tomsk, 634050, Russia
phone: +7 (3822) 41-39-64
irene@gpb.tspace.ru, yevtushenko@elefot.tsu.tomsk.su

‡ *CRIM, Centre de Recherche Informatique de Montréal*
550 Sherbrooke West, Suite 100
Montréal, H3A 1B9, Canada
phone: +1 (514) 840-1234, fax: +1 (514) 840-1244
petrenko@crim.ca

Abstract In this paper, we consider the problem of test derivation from a specification FSM, assuming that all possible implementation FSMs are submachines of some nondeterministic FSM. The latter represents a restricted class of faults defined by the user. The state number in an implementation may exceed that of the specification. We present a method for test generation that can deliver shorter tests than the other existing methods. The method is also more flexible than the traditional FSM-based methods, which embody a universal fault model defined only by a state number.

Keywords: Conformance testing, test generation, fault models, finite state machines.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35567-2_25

G. Csopaki et al. (eds.), *Testing of Communicating Systems*

1. INTRODUCTION

Conformance testing is often formalized as the FSMs equivalence problem. One usually assumes that all possible implementation faults for a given specification machine are described by a certain set of "mutant" FSMs, called a fault domain. The test suite is complete w.r.t. a given fault domain if for each mutant FSM, nonequivalent to the specification FSM, the test suite has a sequence detecting this machine.

In cases when the fault domain is defined only by maximal number of states of implementations under test (a universal fault domain) we can use a number of test derivation methods [Vasi73], [Chow78], [VCI89], [YePe90], [Petr91], [FBKA91], [YaLe95]. The total length of a test suite complete in such a domain exponentially depends on the value (m-n) where m is the maximal number of states of implementation or mutant FSMs, while n is the number of states of the specification FSM. Shorter tests are usually required when the fault domain is further restricted to user-defined faults [GrPe88], [BrJu92], [PeYe92], [PoRe97]. The question is how one can efficiently represent a fault domain that is a proper subset of a universal fault domain.

In cases when it is possible to explicitly enumerate all the FSMs of the fault domain one can determine for each FSM, nonequivalent to the specification FSM, an input sequence distinguishing the two machines to eventually derive a complete test suite [Gill62], [PoMc64]. We do not know, however, whether this technique provides the shortest test suite. In cases, when the number of mutants tends to explode, as it often happens for practical systems, an implicit enumeration of mutants, proposed in [PoRe97], can be attempted. According to this approach, one determines the set of all "smallest" partially specified FSMs and for each machine finds a sequence that distinguishes it from the given specification machine. The approach does not rely on a state cover set and distinguishing sequences of a given specification FSM opposed to classical methods, such as *W, Wp, UIOv, HSI*-methods. By this reason, in the worst case situation, when only the maximal number of states of an implementation FSM is known, total length of tests derived by the method [PoRe97] exponentially depends not on the value (m-n), but on (m+n-1).

A general representation of mutant FSMs based on a so-called *fault function* has been proposed in [GrPe88], [PeYe92]. Existing fault models for restricted faults such as output faults [NaTs81], input faults [Kozl81], component faults of a system of communicating FSMs [GrPe88], [PYD94], "black-box" faults [PYB96], and others correspond to particular forms of the fault function. Methods for test derivation are developed in [GrPe88],

[PeYe92] for the cases when no mutant FSM has more states than a specification FSM.

In this paper, we extent the notion of a fault function to the notion of a mutation machine driven by the observation that faults actually may increase the number of states. In fact, a mutation machine is a nondeterministic FSM such that the set of its submachines is the user-defined fault domain. The number of states of a mutation machine may exceed that of the specification FSM. To elaborate a strategy that provides shorter tests than those derived by classical testing methods we generalize the notion of universal traversal set that is the key construction in the methods, to a family of traversal sets and propose an algorithm for deriving a complete test suite.

The rest of the paper is structured as follows. In Section 2, we give necessary basic notions and definitions. Section 3 demonstrates that a test suite derived by *W*-method can be reduced when the user-defined fault domain is smaller than a universal fault domain defined by the state number. In Section 4, we construct a so-called distinguishing automaton to elaborate a test derivation approach for specification and mutation machines. An algorithm for test derivation is proposed in Section 5.

2. PRELIMINARIES

2.1 Finite State Machines

A *finite state machine* (*FSM*), often simply called a machine throughout this paper, is an initialized (possibly nondeterministic) machine, i. e. 5-tuple $A=(S,X,Y,h,s_0)$, where S is a finite set of n states with $s_0 \in S$ as the initial state, X and Y are finite sets of input and output symbols, respectively, and h is a behavior function, h: $S\times X \to P(S\times Y)$ where $P(S\times Y)$ is a set of all non-empty subsets of $S\times Y$. The machine A is *deterministic* if $|h(s,x)|=1$ for all $(s,x)\in S\times X$; otherwise, it is *nondeterministic*.

For any finite alphabet X, let X^* denote the set of all finite sequences in X containing the empty sequence ε. As usual, we extend the behavior function h of the machine A to a mapping from the set $S\times X^*$ to the set $P(S\times Y^*)$. The function h has two projections h^1 and h^2, usually called the *next state function* and the *output function*. The machine A is said to be *initially connected* or simply *connected* if each state is reachable from the initial state, i.e. for each state $s\in S$ there exists $\alpha\in X^*$ such that $s\in h^1(s_0,\alpha)$.

Given the machine $A=(S,X,Y,h,s_0)$, a machine $B=(S',X,Y,h',s_0)$ is a *submachine* of A if $S' \subseteq S$ and $h'(s,x) \subseteq h(s,x)$ for all $(s,x) \in S' \times X$. The set of all initially connected deterministic submachines of A is denoted $Sub(A)$.

Given two states, s of FSM $A=(S,X,Y,h,s_0)$ and t of FSM $B=(T,X,Y,g,t_0)$, states s and t are said to be *equivalent*, written $s \cong t$, if $g^2(t,\alpha)=h^2(s,\alpha)$ for all input sequences $\alpha \in X^*$, otherwise, states s and t are *distinguishable*, written $s \ncong t$. For each pair of distinguishable states (s,t), there exists an input sequence $\alpha \in X^*$ such that $g^2(t,\alpha) \neq h^2(s,\alpha)$. In this case, the sequence α is said to *distinguish* states s and t. The machine is said to be *reduced*, if any pair of its states is distinguishable.

The machines are *equivalent* if their initial states are equivalent; otherwise, they are *distinguishable*. An input sequence is said to *distinguish* two machines if it distinguishes their initial states.

2.2 Fault Model

Let $A=(S,X,Y,h,s_0)$ be a deterministic reduced connected FSM, called the *specification machine*. We assume that all possible implementations machines are represented by the set of all deterministic submachines of the FSM $M=(T,X,Y,F,t_0)$, called a *mutation machine*. A submachine B of M is called a *nonconforming implementation* if it is not equivalent to A, otherwise, it is called a *conforming implementation*. In other words, we consider the fault model $<A,\cong,Sub(M)>$ [PYB96].

Any finite set $E \subseteq X^*$ of input sequences is a *test suite* w.r.t the fault model $<A,\cong,Sub(M)>$. The test suite E *detects* a nonconforming implementation machine $B \in Sub(M)$ if there exists an input sequence $\alpha \in E$ distinguishing machines A and B. The test suite E is said to be *complete* w.r.t the fault model $<A,\cong,Sub(M)>$ if it detects each nonconforming implementation machine $B \in Sub(M)$.

Next section is devoted to the discussion of possible approaches for deriving tests complete w.r.t. the fault model $<A,\cong,Sub(M)>$.

3. TEST DERIVATION USING EXISTING METHODS

3.1 Universal traversal set

Given an input alphabet X, an output alphabet Y and integer m, there exists a special nondeterministic FSM such that any machine with up to m

states defined over alphabets X and Y is isomorphic to its submachine. In particular, consider a *chaos* machine $Ch_m(X,Y)=(P,X,Y,h,p_0)$, where $|P|=m$ and $h(p,x)=P\times Y$ for all $(p,x)\in P\times X$. The fault model $<A,\cong,Sub(Ch_m(X,Y))>$ is a well known "black-box" model. If a given mutation machine M has m states then a complete test suite w.r.t. $<A,\cong,Sub(Ch_m(X,Y))>$ is also complete w.r.t. $<A,\cong,Sub(M)>$, for any submachine of the mutation machine M is isomorphic to a submachine of $Ch_m(X,Y)$. A number of methods exist for deriving tests complete w.r.t. the fault model $<A,\cong,Sub(Ch_m(X,Y))>$. Below we briefly sketch the W-method [Vasi73], [Chow78].

A finite set W of finite input sequences is called a *characterization set* [Chow78] of a reduced FSM A if for each pair of different states of A, the set W has a sequence, distinguishing the states. Given a finite subset $V \subseteq X^*$ and an integer k, we further denote VX^k the set obtained by concatenating each sequence $\alpha\in V$ with each sequence in X^* up to length k. Let $n(V)$ be a number of distinct states, where sequences of the set V take the FSM A from the initial state, i.e. $n(V)=|\{\delta(s_0,\alpha) \mid \alpha\in V\}|$. The set $VX^{m-|n(V)|+1}W$ is a complete test suite w.r.t. the fault model $<A,\cong,Sub(Ch_m(X,Y))>$.

A	P	Q	R
x	$R/1$	$Q/1$	$P/1$
y	$P/0$	$P/1$	$Q/1$

Figure 1. FSM A.

Example. Consider the machine A (Figure 1) with three states, P, Q, and R; the initial state is P. It has two inputs, x and y; and two outputs, 0 and 1. We apply the W-method to generate tests, assuming that any implementation has at most four states. A characterization set of A is $W=\{yy\}$. Given a set $V=\{\varepsilon,x,xy\}$ of input sequences, $n(V)=3$. We construct the test suite $VX^2W=\{xxxyy,\ xxyyy,\ xyxxyy,\ xyxyyy,\ xyyxyy,\ xyyyyy,\ yxyy,\ yyyy\}$ complete w.r.t. the fault model $<A,\cong,Sub(Ch_4(X,Y))>$.

The set $X^{m-|n(V)|+1}$ that is a key construction of the W-method is called a *universal traversal set* [YaLe95] and enjoys a nice property.

Proposition 3.1. For each nonconforming implementation FSM $B\in Sub(Ch_m(X,Y))$, at least one of the following conditions holds.

- There exist sequences $\alpha\in V$ and $\beta\in X^{m-|n(V)|+1}$ such that the sequence $\alpha\beta$ distinguishes B from A or visits distinguishable states in the FSM

A from the initial state simultaneously visiting one and the same state in the FSM B.

- There exist sequences $\alpha, \gamma \in V$ and $\beta \in X^{m-|n(V)|+1}$ such that the sequences $\alpha\beta$ and γ take A from the initial state to distinguishable states while taking B from the initial state to one the same state.

Since $Ch_m(X,Y)$ is a chaos machine, for any sequences $\alpha, \gamma \in V$, $\beta \in X^{m-|n(V)|+1}$ and any pair of different states of A, there exists $B \in Sub(Ch_m(X,Y))$ such that the conditions of Proposition 3.1 hold. Therefore, to obtain a test suite complete w.r.t. the fault model $<A, \cong, Sub(Ch_m(X,Y))>$ we should concatenate each sequence of the set $VX^{m-|n(V)|+1}$ with a characterization set W. As mentioned above, a test suite complete w.r.t. $<A, \cong, Sub(Ch_m(X,Y))>$ is also complete w.r.t. $<A, \cong, Sub(M)>$ for any mutation FSM M with at most m states. However, in particular cases, when M is a proper submachine of $Ch_m(X,Y)$ the test suite may be reduced without loss of its completeness.

3.2 Test Minimization

Let TS be a test suite complete w.r.t. the fault model $<A, \cong, Sub(Ch_m(X,Y))>$ and $Pref(TS)$ be the set of all prefixes of TS. There may exist a proper subset of $Pref(TS)$ that is complete w.r.t. $<A, \cong, Sub(M)>$. If it is possible to explicitly enumerate all the submachines of the mutation machine M, the subset can be determined as a column coverage of a special Boolean matrix Θ. The item Θ_{ij} of the matrix is '1' iff the sequence α_i of the set $Pref(TS)$ detects a nonconforming submachine B_j of M. The existing methods (see, for example, [John74]) can solve the problem for matrices with thousands rows and columns.

M	1	2	3	4
x	3/1 4/1	3/0,1 4/0,1	1/0,1 2/0,1	4/1
y	1/0,1 4/0,1	4/1	4/1	1/0,1

Figure 2: Mutation FSM M.

Example. Consider the specification machine A with the initial state P shown in Figure 1. Suppose now that all the faulty machines are submachines of the FSM M (Figure 2) with the initial state 1. The entries of this table are interpreted as follows. Consider as an example {1/0,1; 4/0,1}. It means that any implementation in response to input y will transfer to state 1

or 4. In either case, it will produce output 0 or 1. The fault domain consists of 256 machines. By direct inspection, one can assure the set $\{xyy, xxyy\}$ is a column coverage of the corresponding Boolean matrix, i.e. the test suite $\{xyy, xxyy\}$ is complete w.r.t. the fault model $<A,\cong,Sub(M)>$. Its total length is much less than that of a complete test suite w.r.t. $<A,\cong,Sub(Ch_m(X,Y))>$ obtained in the previous section. The example indicates that a complete test suite obtained by the *W*-method may well be redundant when a given mutation machine is a proper submachine of the chaos machine.

In cases when elements of the set $Sub(M)$ cannot be explicitly enumerated, we may use a test strategy based on traversal sets and state identifiers implemented in the *W* or other methods. However, given a mutation machine M, Proposition 3.1 may hold for a proper subset of the set $X^{m-|n(V)|+1}$ when M is a proper submachine of $Ch_m(X,Y)$. As an example, consider a mutation FSM M that has only invalid outputs at the initial state for some input x. In this case, the set $\{x\}$ may serve as a traversal set, for each submachine of M is distinguished from A by x. Thus, for a "non-chaotic" mutation FSM M, the universal traversal set $X^{m-|n(V)|+1}$ may be reduced. Moreover, concatenation of each sequence of the traversal set with a characterization set may also be unnecessary. For some mutation FSMs, as in the above example, we do not need a characterization set at all.

Given a specification FSM A, a mutation FSM M and a finite set V of input sequences, we generalize the notion of universal traversal set to the case when not every FSM with up to m states can be an implementation machine. We extent this notion to a so-called family of M-traversal sets w.r.t. the set V. Necessary properties of traversal sets are established in terms of pairs of states, where an input sequence simultaneously takes specification and implementation machines from their initial states (Proposition 3.1). By this reason, to construct M-traversal sets we use distinguishing automata that are similar to products of FSMs, see e.g. [PoMc64], [ACY95].

4. DISTINGUISHING AUTOMATA

4.1 Finite Automata

A *finite automaton* over the finite alphabet X is a 4-tuple $R=(S,X,\delta,s_0)$, where S is a finite set of states, $s_0 \in S$ is an initial state and δ is a transition

function, δ: $S\times X\rightarrow P(S)$. The automaton R is *deterministic* if $|\delta(s,x)|=1$ for all states $s\in S$ and symbols $x\in X$, otherwise it is *nondeterministic*.

Given an automaton $R=(S,X,\delta,s_0)$, a sequence $\eta*\beta=sx_1s_1x_2s_2\ldots x_ks_k$, where $\eta=ss_1s_2\ldots s_k\in S^*$, $\beta=x_1\ldots x_k\in X^*$, is called a *path* of the automaton R at the state s if $s_1\in\delta(s,x_1)$ and $s_i\in\delta(s_{i-1},x_i)$ for all i, $1<i$ k; the sequence β is called *X-projection* of the path $\eta*\beta$. Given a sequence $\beta\in X^*$, any path $\eta*\beta$ at the initial state of R is said to be *created* by the sequence β in the automaton R. As usual, the integer k is called *length* of the path, while states s and s_k are called the *head* and the *tail* states of the path. A path at the initial state of the automaton is simply called a *path of the automaton* throughout this paper. When we want to indicate that the head state of the path $\eta*\beta$ is s we rewrite it as $s\eta*\beta$. For each state s of the automaton R, there always exists a special path $ss*\varepsilon$, written $s*\varepsilon$ for short, where ε is the empty sequence. By definition, this path has zero length and its head and tail states coincide. A path $\eta*\beta$ is said to *traverse* state $s\in S$ if the sequence η includes state s . State s of the automaton R is called *reachable* if there exists a path of R traversing the state s.

Given the automaton $R=(S,X,\delta,s_0)$, an automaton $R=(S,X,\delta,s_0)$ is a *sub-automaton* of R if $S\subseteq S$ and $\delta(s,x)\subseteq\delta(s,x)$ for all $(s,x)\in S\times X$.

4.2 Distinguishing Automaton of Two FSMs

Given two FSMs $A=(S,X,Y,h,s_0)$ and $B=(T,X,Y,g,t_0)$, we construct an automaton $((S\times T)\ \{Fail\},X,f,s_0t_0)$, such that for all $(s,t)\in S\times T$ and all $x\in X$ it holds that:

- $f(st,x)=\begin{cases}\Sigma(st,x), \text{ if } h^2(s,x)=g^2(t,x)\\ \Sigma(st,x)\ \ \{Fail\}, \text{ if } h^2(s,x)\ \ g^2(t,x)\end{cases}$

where $\Sigma(st,x)=\{s\ t\ \mid\ \exists y\in h^2(s,x)\cap g^2(t,x)\ \ [(s,y)\in h(s,x)\ \ \&\ (t,y)\in g(t,x)]\}$;

- $f(Fail,x)=\{Fail\}$.

Let Q denote the set of all reachable states of the automaton. Then the sub-automaton (Q,X,f,q_0), where $q_0=s_0t_0$, is called the *distinguishing automaton*, denoted $A\ \ B$. In this definition, *Fail* is a designated state which indicates that the two machines cannot agree on outputs.

A M	$P1$	$P2$	$P4$	$Q1$	$Q2$	$Q3$	$Q4$	$R3$	$R4$	*Fail*
x	$R3$ $R4$	$R3$ $R4$ *Fail*	$R4$	$Q3$ $Q4$	$Q3$ $Q4$ *Fail*	$Q1$ $Q2$ *Fail*	$Q4$	$P1$ $P2$ *Fail*	$P4$	*Fail*
y	$P1$ $P4$ *Fail*	*Fail*	$P1$ *Fail*	$P1$ $P4$ *Fail*	$P4$	$P4$	$P1$ *Fail*	$Q4$	$Q1$ *Fail*	*Fail*

Figure 3: The distinguishing automaton A M.

Example. The distinguishing automaton A M for the machines A (Figure 1) and M (Figure 2) with the initial states P and 1, respectively, is shown in Figure 3. The initial state of A M is $P1$.

For deterministic machines A and B, the distinguishing automaton A B is often used to verify whether machines A and B are equivalent. In fact, the automaton A B has the state *Fail* if and only if the machines A and B are distinguishable. The language accepted by the state *Fail* is the set of all sequences distinguishing A and B.

Proposition 4.1. Given a deterministic FSM A and an FSM M over the same input and output alphabets, let A M be the distinguishing automaton for A and M. For any machine $B \in Sub(M)$, the distinguishing automaton A B is a sub-automaton of A M.

Corollary 4.2. Given a distinguishing automaton A B, $B \in Sub(M)$, let $\eta * \beta$ be a path of A B. Then $\eta * \beta$ is also a path of the distinguishing automaton A M.

For any state $q=st$ of the distinguishing automaton A M, the states s and t of machines A and M are S-projection and T-projection of q, respectively. Two paths $\sigma * \alpha$ and $\rho * \gamma$ at arbitrary states of the distinguishing automaton A M are said to be *compatible*, if for any two segments qxq of $\sigma * \alpha$ and pxp of $\rho * \gamma$ such that states q and p have the same T-projection, either one of states q and p is *Fail*, or the T-projections of states q and p also coincide. Path $\sigma * \alpha$ at an arbitrary state of the distinguishing automaton A M is called *deterministic* if it is compatible with itself.

Proposition 4.3. For any implementation $B \in Sub(M)$, each path of the distinguishing automaton A B is deterministic and any two paths of A B are compatible.

4.3 Detecting Nonconforming Machines

In this section, we introduce a notion of a sequence set that "traps" a nonconforming implementation $B \in Sub(M)$ just as the set $VX^{m-|n(V)|+1}$ does for machines in the set $Sub(Ch_m(X,Y)$ (Proposition 3.1). Due to Corollary 4.2, pairs of states where input sequences simultaneously take a specification machine A and an implementation $B \in Sub(M)$, are states of the distinguishing automaton A M; thus, we need to study properties of the states first.

Let A $M=(Q,X,H,q_0)$ be the distinguishing automaton of FSMs A and M, and q *Fail* be a state of A M. The state q of A M is 1-*forbidden* if there exists a symbol $x \in X$ such that $H(q,x)=\{Fail\}$. Suppose we have determined all $(k-1)$-forbidden states of the automaton A M, $k>1$. The state q of A M is *k-forbidden* if it is $(k-1)$-forbidden or there exists a symbol $x \in X$ such that each state of the set $H(q,x)$ is $(k-1)$-forbidden.

Since the set Q of states of the distinguishing automaton A M is finite there exists an integer k such that the sets of k-forbidden and $(k+1)$-forbidden states coincide, i.e. the procedure is exhaustive. We call a state *forbidden* if there exists an integer k such that it is k-forbidden. As an example, state $P2$ of the distinguishing automaton A M (Figure 3) is forbidden.

Proposition 4.4. If the initial state of the automaton A M is forbidden then each implementation machine $B \in Sub(M)$ is nonconforming.

For each forbidden state q of A M, there exists a so-called *distinguishing set* $W(q)$ such that for any sub-automaton A B, $B \in Sub(M)$, with the state q, at least one sequence $\alpha \in W(q)$ takes A B from the state q to the state *Fail*. A set $W(q)$ can be constructed as follows.

Let q be an 1-forbidden state of A $M=(Q,X,H,q_0\}$. Then there exists a symbol $x \in X$ such that $H(q,x)=\{Fail\}$. The set $\{x\}$ may serve a distinguishing set $W(q)$. Suppose we have determined distinguishing sets for all $(k-1)$-forbidden states of the automaton A M, k 1, and state q is k-forbidden. If q is $(k-1)$-forbidden then its distinguishing set is determined. Otherwise, there exists a symbol $x \in X$ such that each state of the set $H(q,x)$ is $(k-1)$-forbidden. The union $W(q)$ of the sets $xW(q\ )$ over all states $q \in H(q,x)$ can serve a distinguishing set for the state q. As an example, the forbidden state $P2$ has a distinguishing set $W(P2)=\{y\}$.

By definition, if a distinguishing automaton A B, $B \in Sub(M)$, has a forbidden state then it has the state *Fail*. We also introduce a notion of

conflicting states in the automaton A M. Any distinguishing automaton A B with conflicting states also has the state *Fail*.

Given a specification FSM A and a mutation FSM M, two states s_1t and s_2t for different states s_1 and s_2 of A and some state t of M are called *conflicting* states. The states s_1t and s_2t are said to *conflict* on the sequence ω if the sequence ω distinguishes states s_1 and s_2 in the FSM A. For any distinguishing automaton A B with states s_1t and s_2t, the sequence ω takes A B to the state *Fail* from at least one of states s_1t and s_2t.

A set $L \subseteq X^*$ of sequences is said to *trap* a machine $B \in Sub(M)$ (w.r.t. the specification machine A), if the set of tail states of paths of A B created by the sequences in L has two conflicting states or a state that is *Fail* or forbidden. Given a set $L_B \subseteq X^*$ that traps B, we can easily construct a set of sequences distinguishing B and A. In fact, if there exists a path $\eta * \beta$, $\beta \in L_B$, of A B, whose tail state is *Fail* then the sequence β distinguishes machines A and B. If some path $\eta * \beta$, $\beta \in L_B$, of A B has a forbidden tail state q then at least one sequence of the set $\beta W(q)$, where $W(q)$ is a distinguishing set of the state q, distinguishes A and B. Finally, if A B has two paths $\eta_1 * \beta_1$, $\eta_2 * \beta_2$, $\beta_1, \beta_2 \in L_B$, such that their tail states conflict, then one of sequences $\beta_1\omega_{12}$ and $\beta_2\omega_{12}$, where ω_{12} is a sequence on which the states conflict, distinguishes A and B.

A path $\eta * \beta$ of A M is called *nonconforming* if η includes two conflicting states or a state that is *Fail* or forbidden; otherwise the path is *conforming*. Given an implementation FSM B, $B \in Sub(M)$, if the distinguishing automaton A B has a nonconforming path then B is a nonconforming implementation and the set $Pref(\beta)$ of all prefixes of the sequence β traps B.

5. CONSTRUCTING TESTS BASED ON DISTINGUISHING AUTOMATA

5.1 Algorithm for Test Derivation

In this section, we develop the notion of a family of M-traversal sets by generalizing universal traversal sets to incorporate the mutation machine M. A family of M-traversal sets serves a basis for deriving tests complete w.r.t. the fault model $<A, \cong, Sub(M)>$.

First, we give some additional notations. For any sequence $\alpha \in X^*$, we use $Path(\alpha)$ to denote the set of all deterministic paths of the distinguishing automaton A M created by α. Given a sequence α and a deterministic path

$\rho * \gamma$ of $A \quad M$, we denote $Path(\alpha)|(\rho * \gamma)$ the set of all paths of $Path(\alpha)$ compatible with $\rho * \gamma$. For any set $V \subseteq X^*$, let $Path(V)$ $(Path(V)|(\rho * \gamma))$ denote the union of the sets $Path(\alpha)$ $(Path(\alpha)|(\rho * \gamma))$ over all $\alpha \in V$.

Definition 5.1. Given the distinguishing automaton $A \quad M$ of FSMs A and M, a finite set V of sequences and the set $Path(V)$ of all paths of $A \quad M$ created by V, let $\mathfrak{I}$ be a family of sets $Tr(\sigma * \alpha)$ of paths $q\eta * \beta$ at the tail state q of $\sigma * \alpha$, for all $\sigma * \alpha \in Path(V)$. The family $\mathfrak{I}$ is said to be a *family of M-traversal sets* w.r.t. the set V if for each nonconforming implementation $B \in Sub(M)$, there exist paths $\sigma * \alpha \in Path(V)$ and $q\eta * \beta \in Tr(\sigma * \alpha)$ such that $\sigma\eta * \alpha\beta$ is a path of the distinguishing automaton $A \quad B$ and the set $Pref(\alpha\beta)$ or the set $V \quad \{\alpha\beta\}$ traps the machine B w.r.t. A.

Any mutation machine with up to m states is a submachine of the FSM $Ch_m(X,Y)$. Thus, due to Proposition 3.1, a family of the sets $Tr(\sigma * \alpha)$, where for each $\sigma * \alpha \in Path(V)$, the set $Tr(\sigma * \alpha)$ comprises all paths of length up to m-$n(V)$+1 at the tail state of $\sigma * \alpha$, is a family of M-traversal set w.r.t. V. However, for a "non-chaotic" machine M, such M-traversal sets may be redundant as it was demonstrated in Section 3.2.

Based on the notion of a family of M-traversal sets, we propose a method for deriving a test suite complete w.r.t. the fault model $<A, \cong, Sub(M)>$. We further assume that the set $Sub(M)$ has at least one nonconforming implementation; otherwise no implementation needs any test at all.

Algorithm 5.1.

Input: The distinguishing automaton $A \quad M$, the set of all forbidden states of $A \quad M$ with their distinguishing sets, the set of all pairs of conflicting states of $A \quad M$, for each pair of conflicting states, a sequence on which the states conflict, a finite set $V \quad X^*$ and a family of M-traversal sets $Tr(\sigma * \alpha)$, $\sigma * \alpha \in Path(V)$, w.r.t. the set V.

Output: A complete test suite w.r.t. the fault model $<A, \cong, Sub(M)>$.

Method. Step 1. For each $\alpha \in V$, construct the set $E_\alpha \quad X^*$ of sequences as follows. Consider each deterministic path $\sigma\eta * \alpha\beta$, where $\sigma * \alpha \in Path(V)$ and $q\eta * \beta \in Tr(\sigma * \alpha)$.

- If the path $\sigma\eta * \alpha\beta$ is nonconforming, determine its shortest nonconforming prefix $\eta * \beta$; let p denote the tail state of $\eta * \beta$. If $p = Fail$ then include β into the set E_α. If p is forbidden then include into the set E_α each sequence of the set $\beta \; W(p)$, where $W(p)$ is a distinguishing set of the state p. If p is neither forbidden state nor *Fail*, determine a prefix $\eta * \beta$ of $\eta * \beta$, whose tail state conflicts with the

state p and include into the set E_α the sequences $\beta\ \omega$ and $\beta\ \omega$, where ω is the sequence, on which the states conflict.

- If the path $\sigma\eta\cdot\alpha\beta$ is conforming, then for each pair of compatible conforming paths $\eta\cdot\beta$, $\eta\cdot\beta \in Path(V)|(\sigma\eta\cdot\alpha\beta)$ $\{\sigma\eta\cdot\alpha\beta\}$, whose tail states conflict, include into the set E_α the sequences $\beta\ \omega$ and $\beta\ \omega$, where ω is a sequence on which the states conflict.

Step 2. Find the set TS as union of sets E_α over $\alpha \in V$.

❑

The following theorem immediately results from Algorithm 5.1 and the definition of a family of M-traversal sets.

Theorem 5.1. Given the distinguishing automaton $A\ M$ of a specification machine A and a mutation machine M, and a finite set $V\ X^*$ of sequences, let a family of sets $Tr(\sigma\cdot\alpha)$, $\sigma\cdot\alpha \in Path(V)$, of paths of $A\ M$ be a family of M-traversal sets w.r.t. V. Then the set TS derived by Algorithm 5.1 is a test suite complete w.r.t. the fault model $\langle A, \cong, Sub(M)\rangle$.

❑

Total length of a test suite delivered by Algorithm 5.1 essentially depends on total length of a given family of M-traversal sets. If the distinguishing automaton $A\ M$ has no forbidden states and the sets of X-projections of M-traversal sets of a given family are proper subsets of the set $X^{m-|n(V)|+1}$ then Algorithm 5.1 provides tests shorter than W-method. In the next section, given a distinguishing automaton $A\ M$, we present an algorithm for deriving a family of M-traversal sets.

5.2 Constructing M-traversal sets

Given a distinguishing automaton $A\ M$ and a finite set $V\ X^*$, as mentioned in Section 5.1, the family of all paths of length up to $m-|n(V)|+1$ at tail states of the paths $\sigma\cdot\alpha \in Path(V)$ is a family of M-traversal sets w.r.t. the set V. Moreover, for certain mutation machines this length can be reduced as follows.

For any sequence $\alpha \in X^*$ and any path $\rho\cdot\gamma$ at the initial state of the distinguishing automaton $A\ M$, let $P(\alpha)$ (or $P(\alpha, \rho\cdot\gamma)$) denote the set of tail states of all conforming paths of the set $Path(\alpha)$ (or $Path(\alpha)|(\rho\cdot\gamma)$, respectively). As an example, for the distinguishing automaton $A\ M$ (Figure 3), the sequence $xyxx$ and the path $P1xR3yQ4yP1xR3$, we have $P(xyxx, P1xR3yQ4yP1xR3) = \{R3\}$.

Proposition 5.2. Given the distinguishing automaton $A\ M$ and a set V of sequences such that the sets $P(\alpha)$, $\alpha \in V$, are pairwise disjoint. A family

of the sets $Tr(\sigma_* \alpha)$, $\sigma_* \alpha \in Path(V)$, where $Tr(\sigma_* \alpha)$ is the set of all paths of length up to $m-|V|+1$ at the tail state of the path $\sigma_* \alpha$, is a family of M-traversal set w.r.t. V.

Given an arbitrary set $V \subseteq X^*$, let $\hat{V}$ denote a maximal subset of V such that the subsets $P(\alpha)$, $\alpha \in \hat{V}$, are pairwise disjoint. Then, due to Proposition 5.2, the family of all paths of length up to $m-|\hat{V}|+1$ at tail states of paths $\sigma_* \alpha \in Path(\hat{V})$ is a family of M-traversal sets w.r.t. the set $\hat{V}$ and, since $\hat{V}$ is a subset of V, is a family of M-traversal sets w.r.t. the set V, as well. Obviously, cardinality of the set $\hat{V}$ is not less than the value of $n(V)$, and thus, total length of a family of M-traversal sets of Proposition 5.2 is not more than that stated in Proposition 3.1.

Due to these considerations, to construct a family of M-traversal sets w.r.t. V, we may determine the maximal subset $\hat{V}$ with the above property, and construct a family of M-traversal sets w.r.t. $\hat{V}$. Moreover, considering each particular path we may further reduce the family. Given the distinguishing automaton of machines A and M, and a set $V \subseteq X^*$ such that the subsets $P(\alpha)$, $\alpha \in V$, are pairwise disjoint, we now present an algorithm for deriving a family of M-traversal sets w.r.t. V.

Algorithm 5.2.

Input: The distinguishing automaton $A \times M$ and a finite set $V \subseteq X^*$ with the empty sequence ε, such that the sets $P(\alpha)$, $\alpha \in V$, are pairwise disjoint.

Output: A family of M-traversal sets $Tr(\sigma_* \alpha)$, $\sigma_* \alpha \in Path(V)$, w.r.t. V.

Method. Step 1. For each deterministic nonconforming path $\sigma_* \alpha \in Path(V)$ with the tail state q, assign $Tr(\sigma_* \alpha) = \{q_* \varepsilon\}$.

Step 2. For each deterministic conforming path $\sigma_* \alpha \in Path(V)$ construct the set $Tr(\sigma_* \alpha)$ as follows. If there exists a proper prefix of the path $\sigma_* \alpha$ that is in the set $Path(V)$ and has the tail state q, where q is the tail state of $\sigma_* \alpha$, then assign $Tr(\sigma_* \alpha) =$. Otherwise, consider each deterministic path $q \eta_* \beta$ of length $m-|V|+1$ at the state q of $A \times M$.

- If the path $\sigma\eta_* \alpha\beta$ is conforming, the states of the sequence η are distinct and for each $\gamma \in V$, the set $P(\gamma, \sigma\eta_* \alpha\beta)$ is not empty and is not a subset of states of η, then include into the set $Tr(\sigma_* \alpha)$ each prefix of $q \eta_* \beta$.
- If the path $\sigma\eta_* \alpha\beta$ is nonconforming then determine the shortest prefix $q\eta_* \beta$ of $q\eta_* \beta$ such that the path $\sigma\eta_* \alpha\beta$ is still nonconforming. If the states of the sequence η are distinct and for

each $\gamma \in V$, the set $P(\gamma, \sigma\eta * \alpha\beta)$ is not empty and is not a subset of states of η, then include the path $q\eta * \beta$ into the set $Tr(\sigma * \alpha)$.

❑

Theorem 5.3. Given the distinguishing automaton A M and a finite set V X^* such that $\varepsilon \in V$ and the sets $P(\alpha)$, $\alpha \in V$, are pairwise disjoint, a family of subsets $Tr(\sigma * \alpha)$ of paths of A M derived by Algorithm 5.2 is a family of M-traversal sets w.r.t. V.

Proof. Consider a nonconforming implementation $B \in Sub(M)$. Due to Proposition 4.1, the distinguishing automaton A B is a sub-automaton of A M with the state *Fail*. Thus, each path at the initial state of A B created by a sequence of the set V is a deterministic path of the set $Path(V)$. Let $P_{A\ B}(V)$ denote the set of tail states of all paths, where sequences of V take A B from the initial state.

If for some sequence $\alpha \in V$ the path $\sigma * \alpha$ at the initial state of A B is nonconforming then, due to Step 1, there exist paths $\sigma * \alpha \in Path(V)$ and $q * \varepsilon \in Tr(\sigma * \alpha)$ such that $\sigma * \alpha\varepsilon$ is a path of the distinguishing automaton A B and the set $Pref(\alpha\varepsilon)$ traps the machine B w.r.t. the specification machine A.

Suppose now each path $\sigma * \alpha$, $\alpha \in V$, at the initial state of A B is conforming, i.e. *Fail* $P_{A\ B}(V)$. Let the path $q\chi * \delta$, $q \in P_{A\ B}(V)$, be the shortest one of all paths at states of the set $P_{A\ B}(V)$ such that their tail state is *Fail*; and $\sigma * \alpha$ be the shortest path at the initial state of A B such that $\alpha \in V$ and the tail state of $\sigma * \alpha$ is q. Obviously, states of the sequence χ are distinct and the sequence χ does not contain the states of the set $P_{A\ B}(V)$. This means that for each $\gamma \in V$, the set $P(\gamma, \sigma\chi * \alpha\delta)$ is not empty and is not a subset of states of χ; and the same holds for each proper prefix of $q\chi * \delta$. Since $\sigma * \alpha$ is the shortest path of A B with the tail state q, such that $\alpha \in V$, each proper prefix of the path $\sigma * \alpha$ that is in the set $Path(V)$ has the tail state different from q; thus, the set $Tr(\sigma * \alpha)$ is not empty.

Suppose the path $q\chi * \delta$ has prefixes of length of up to $m-|V|+1$ that constitute nonconforming paths when concatenated to $\sigma * \alpha$. Then let $q\eta * \beta$ denote the shortest such prefix, i.e. the path $\sigma\eta * \alpha\beta$ is nonconforming. Due to Step 2, $q\eta * \beta \in Tr(\sigma * \alpha)$. Thus, there exist paths $\sigma * \alpha \in Path(V)$ and $q\eta * \beta \in Tr(\sigma * \alpha)$ such that $\sigma\eta * \alpha\beta$ is a path of the distinguishing automaton A B and the set $Pref(\alpha\beta)$ traps the machine B.

Suppose that for any prefix $q\eta * \beta$ of length of up to $m-|V|+1$ of the path $q\chi * \delta$, the path $\sigma\eta * \alpha\beta$ is conforming. This means that length of $q\chi * \delta$ exceeds $m-|V|+1$. In this case, let $q\eta * \beta$, $\eta = q_1 q_2 \ldots q_k$, denote a prefix of the path $q\chi * \delta$ of length $k = m-|V|+1$. Due to Step 2, each prefix of $q\eta * \beta$ is in the set $Tr(\sigma * \alpha)$. Since the sets $P(\alpha)$, $\alpha \in V$, are pairwise disjoint, it holds that $|P_{A\ B}(V)| = |V|$; thus, the cardinality of the set $P_{A\ B}(V)$ $\{q_1, \ldots, q_k\}$ is

$m+1$. The states of the set $P_{A\ B}(V)$ $\{q_1,\ldots, q_k\}$ are distinct; therefore, there are two conflicting states in the set. This means, that there exists a prefix $q\eta *\beta \in Tr(\sigma * \alpha)$ of $q\eta * \beta$ such that the set V $\{\alpha\beta\}$ traps B w.r.t. A. Thus the family of sets derived by Algorithm 5.2 is a family of M-traversal sets w.r.t. V.

❑

Example. Consider the distinguishing automaton A M in Figure 3 and the set $V=\{\varepsilon,x,xy\}$ of sequences that is a state cover of A. The path $P1xR4yQ1 \in Path(V)$ is nonconforming, thus, $P(xy)=\{Q4\}$. By direct inspection, one can assure that the sets $P(\varepsilon)=\{P1\}$, $P(x)=\{R3,\ R4\}$, $P(xy)=\{Q4\}$ are pairwise disjoint, i.e. $\hat{V}=V$ and length of a path of any M-traversal set derived by Algorithm 5.2 does not exceed $m-|V|+1=2$. We illustrate Algorithm 5.2 constructing the set $Tr(P1xR3yQ4)$ of paths at the tail state $Q4$ of the path $P1xR3yQ4$. Consider all paths of length 2 at the state $Q4$, i.e. the paths (1) $Q4xQ4xQ4$, (2) $Q4xQ4yP1$, (3) $Q4xQ4yFail$, (4) $Q4yP1xR3$, (5) $Q4yP1xR4$, (6) $Q4yP1yP1$, (7) $Q4yP1yP4$, (8) $Q4yP1yFail$, (9) $Q4yFailxFail$, (10) $Q4yFailyFail$.

(1), (6): the paths $P1xR3yQ4xQ4xQ4$ and $P1xR3yQ4yP1yP1$ are conforming and the states of the sequences $Q4Q4$ and $P1P1$, respectively, are not distinct. In this case, we do nothing.

(2), (4): the paths $P1xR3yQ4xQ4yP1$ and $P1xR3yQ4yP1xR3$ are conforming and the sets $P(xy,P1xR3yQ4xQ4yP1)=0Q4\}$ and $P(\varepsilon,P1xR3yQ4yP1xR3)=\{P1\}$ are subsets of states of the sequences $Q4P1$ and $P1R3$, respectively. We do nothing.

(3), (5), (7), (8): the paths $P1xR3yQ4xQ4yFail$, $P1xR3yQ4yP1xR4$, $P1xR3yQ4yP1yP4$, $P1xR3yQ4yP1yFail$ are nonconforming while all their proper prefixes are conforming. The sets $P(xy,P1xR3yQ4xQ4yFail)=\{Q4\}$, $P(\varepsilon,P1xR3yQ4yP1xR4)=\{P1\}$, $P(\varepsilon,P1xR3yQ4yP1yP4)=\{P1\}$, $P(\varepsilon,P1xR3yQ4yP1yFail)=\{P1\}$ are subsets of states of the sequences $Q4Fail$, $P1R4$, $P1P4$, $P1Fail$, respectively. We do nothing.

(9), (10): the paths $P1xR3yQ4yFailxFail$ and $P1xR3yQ4yFailyFail$ are nonconforming and $Q4yFail$ is the shortest prefix of $Q4yFailxFail$ and $Q4yFailyFail$ such that the path $P1xR3yQ4yFail$ is also nonconforming. The states of the sequence $Q4Fail$ are distinct and for any sequence $\gamma \in V$ the set $P(\gamma,P1xR3yQ4yFail)$ is not empty and is not a subset of states of $Q4Fail$. We include the path $Q4yFail$ into the set $Tr(P1xR3yQ4)$.

Thus $Tr(P1xR3yQ4)=\{Q4yFail\}$. Algorithm 5.2 results in the following family of M-traversal sets:

$Tr(P1*\varepsilon)=\{\ P1yP4xR4, P1yP4yFail, P1yFail\ \}$;

$Tr(P1xR3)=\{\ R3xP2, R3xFail\ \}$;

$Tr(P1xR4)=$;

$Tr(P1xR3yQ4) = \{ Q4yFail \}$;
$Tr(P1xR4yQ1) = \{ Q1 * \varepsilon \}$;
$Tr(P1xR4yFail) = \{ Fail * \varepsilon \}$.

We now call Algorithm 5.1 to derive a test suite complete w.r.t. the fault model $<A, \cong, Sub(M)>$. The states P and Q, as well as the states P and R, are distinguished by the sequence y. The sequence yy may serve as a distinguishing sequence for Q and R. At Step 1 of the algorithm, we obtain $E_\varepsilon = \{yxy, yy\}$, $E_x = \{xxy\}$, $E_{xy} = \{xyy, y\}$. Due to Theorem 5.3, the union $E = \{xxy, xyy, yxy, yy\}$ of the sets E_ε, E_x and E_{xy} is a test suite complete w.r.t. $<A, \cong, Sub(M)>$.

The Algorithm 5.2 usually delivers a shorter family of M-traversal sets than that created by all sequences of length up to $m-|V|+1$. However, the resulting family can further be reduced if one undertakes a more complicated analysis of the obtained paths. For example, consider the traversal set $Tr(P1 * \varepsilon) = \{P1yP4xR4, P1yP4yFail, P1yFail\}$. In fact, for each implementation $B \in Sub(M)$ such that the distinguishing automaton $A \quad M$ has the path $P1yP4$, the sequence $xy \in V$ either creates a nonconforming path in $A \quad M$, or takes $A \quad M$ from the initial state to the state $Q4$ conflicting with the tail state of $P1yP4$. Thus, replacing the set $Tr(P1 * \varepsilon)$ with the set $Tr(P1 * \varepsilon) = \{P1yP4, P1yFail\}$, we obtain a shorter family that is also a family of M-traversal sets. Applying Algorithm 5.1, we derive a set $E = \{xxy, xyy, yy\}$ that is a test suite complete w.r.t. $<A, \cong, Sub(M)>$. This example indicates that more research is needed to optimise the family of M-traversal sets and eventually a complete test suite.

6. CONCLUSION

In this paper, we presented a test derivation approach to deal with situations when the specification FSM is a reduced completely specified FSM, while any implementation machine is a deterministic submachine of some nondeterministic FSM, called a mutation FSM. The mutation machine allows the user to model faults in a very compact way. Compared to the previous work based on a fault function, we allow for an implementation to have more states than its specification. Similar to other existing methods, the key step in our method is construction of traversal sets. We extended the notion of universal traversal set to a so-called family of M-traversal sets and based on this new notion, we proposed a method for test generation that can deliver shorter (at least never longer) tests than the other existing methods. The method is also more flexible than the traditional FSM-based methods, which

embody a universal fault model defined only by the state number. Our future work is related to generalization of the proposed approach to nondeterministic partially defined specification and weakly initialized mutation FSMs.

Acknowledgments. This work was in part supported by the NSERC grant OGP0194381.

REFERENCES

[ACY95] R. Alur, C. Courcoubetis, and M. Yannakakis, *Distinguishing tests for nondeterministic and probabilistic machines*, Proceedings of the 27th ACM Symposium on Theory of Computing, 1995, pp. 363-372.

[BrJu92] J. A. Brzozowski, H. Jurgensen, *A Model for sequential machine testing and diagnosis*, Journal of Electronic Testing: Theory and Applications, 2, 1992, pp. 219-234.

[Chow78] T. S. Chow, *Test software design modeled by finite state machines*, IEEE Transactions, SE-4, No. 3, 1978, pp. 178-187.

[FBKA91] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, A. Ghedamsi, *Test selection based on finite state models*, IEEE Trans. SE-17, No. 6, 1991, pp. 591-603.

[Gill62] A. Gill, *Introduction to the theory of finite-state machines*, McGraw-Hill, 1962, 207 p.

[GrPe88] I. Grunskij, A. Petrenko, *Design of checking experiments with automata describing protocols*, in Automatic Control and Computer Science, Allerton Press Inc., N.Y., 1988, No.4, pp. 7-14.

[John74] D. S. Johnson, *Approximation algorithms for combinatorial problems*, Journal of Computer and System Sciences, 9, 1974, pp. 256-278.

[Kozl81] V. A. Kozlovsky, *On the recognition of an automaton relative to a locally generated class*, Soviet Math. Dokl., 1981, vol.23, No.3, pp. 625-628.

[NaTs81] S. Naito and M. Tsunoyama, *Fault detection for sequential machines by transition tours*, Proceedings of Fault Tolerant Comp. Syst., 1981, pp. 238-243.

[Petr91] A. Petrenko, *Checking experiments with protocol machines*, IFIP Transactions, Protocol Testing Systems IV (the Proceedings if IFIP TC6 4th International Workshop on Protocol Test Systems, 1991), Ed. by Jan Kroon, Rudolf J. Heijink and Ed Brinksma, 1992, North-Holland, pp. 83-94.

[PeYe92] A. Petrenko, N. Yevtushenko, *Test suite generation for a given type of implementation errors*, Proceedings of the IFIP XII International Conference on Protocol Specification, Testing and Verification, 1992, pp. 229-243.

[PYD94] A. Petrenko, N. Yevtushenko, and R. Dssouli, *Testing strategies for communicating FSMs,* Proceedings of the 7th IWTCS, 1994, pp. 193-208.

[PYB96] A. Petrenko, N. Yevtushenko, G. v. Bochmann, and R. Dssouli. *Testing in context: framework and test derivation*, Computer communications, Vol. 19, pp. 1236-1249, 1996.

[PoMc64] J. F. Poage, E. J. McCluskey, *Derivation of optimum test sequences for sequential machines*, Proceedings of the 5th Annu. Symp. On Switching Theory and Logical Design, 1964.

[PoRe97] I. Pomeranz, S. M. Reddy, *Test generation for multiple state-table faults in finite-state machines*, IEEE Transactions on Computers, Vol. 46, No 7, pp. 782-794, 1997.

[Vasi73] M. P. Vasilevsky, *Failure diagnosis of automata*, Cybernetics, Plenum Publishing Corporation, NY, No. 4, 1973, pp. 653-665.

[VCI89] S. T. Vuong, W. W. L. Chan, M. R. Ito, *The UIO-method for protocol test sequence generation*, Proceedings of the 2nd IFIP International Workshop on Protocol Test Systems, 1989, pp. 161-175.

[YaLe95] M. Yannakakis, D. Lee, *Testing finite state machines: fault detection*, Journal of Computer and System Sciences, 1995, 50, pp. 209-227.

[YePe90] N. Yevtushenko, A. Petrenko, *A method of constructing a test experiment for an arbitrary deterministic automaton*, Automatic Control and Computer Science, Allerton Press Inc., N.Y., 1990, Vol. 24, No. 5, pp.65-68.

BIOGRAPHY

Irina Koufareva received the Dipl. degree in computer sciences in 1994 from Tomsk State University, Russia. At the moment, she is an assistant professor at that university. Her current research interests include the automata theory and formal methods in conformance testing.

Alexandre Petrenko received the Dipl. degree in electrical and computer engineering from Riga Polytechnic Institute and the Ph.D. degree in computer science from the Institute of Electronics and Computer Science, Riga, USSR. In 1996, he has joined CRIM, Centre de Recherche Informatique de Montréal, Canada. He is also an adjunct professor of the Université de Montréal, where he was a visiting professor / researcher from 1992 to 1996. From 1982 to 1992, he was the head of a research department of the Institute of Electronics and Computer Science in Riga. From 1979 to 1982, he was with the Networking Task Force of the International Institute for Applied System Analysis (IIASA), Vienna, Austria. His current research interests include high-speed networks, communication software engineering, formal methods, conformance testing, and testability.

Nina Yevtushenko received the Dipl. degree in radio-physics in 1971 and Ph.D. degree in computer sciences in 1983, both from Tomsk State University. She is currently a professor at that University. Her research interests include the automata and FSM theory and testing problems