6

Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines[1]

Gang Luo[a], Alexandre Petrenko[b] and Gregor v. Bochmann[b]

[a] Bell Northern Research, Ottawa, Ontario, Canada K1Y 4H7.

[b] Departement d'IRO, Universite de Montreal, C.P. 6128, Succ.Centre-ville, Montreal, P.Q., H3C 3J7, Canada

In order to test the control portion of communication software, specifications are usually first abstracted to state machines, then test cases are generated from the resulting machines. The state machines obtained from the specification are often *both* partially-specified *and* nondeterministic, but no former work has been reported on test generation for such state machines. We come out with a method of generating test suites for the software that is modeled by partially-specified nondeterministic finite state machines (PNFSM's). On the basis of intuitive notions, a conformance relation, called *quasi-equivalence*, is introduced for such machines, which serves as a guide to test generation. Our method is also applicable to completely-specified deterministic machines, partially-specified deterministic machines, and completely-specified nondeterministic machines, which are typical classes of PNFSM's. When applied to such classes of machines, this method yields not greater test suites with full fault coverage for each class of machines than the existing methods for the same class which also provide full fault coverage, when the number of states in implementation machines is bounded by a known integer. The test suites generated by the method can be used to check the conformance relation between a specification and its implementations.

Key Word Codes: C.2.2: D.2.5
Key Words: Network Protocols; Testing and Debugging

1. INTRODUCTION

The testing phase represents a large effort within the common software development cycle. In the area of communication software, systematic approaches have been developed for protocol conformance testing [37], and the selection of appropriate test suites [14, 36, 42, 41, 40, 10, 34]. These approaches can produce significant economic benefits [1, 2]. Usually, the specifications of communication software are first abstracted to state machines, then test cases are generated from the resulting machines [25, 38]. A considerable amount of work has been done to generate test cases for *completely-specified, deterministic* finite state machines (FSM's) [14, 42, 10, 18, 47, 39, 32, 46]. However, the specifications of communication software often contain *both* nondeterministic *and* partially (or incompletely) specified behavior. For example, all the three major specification languages for communication software, LOTOS [3, 22], ESTELLE [8, 21] and SDL [4] support the description of nondeterminism; and ESTELLE

[1] This work was supported by the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols at the University of Montreal (Canada)

T. Mizuno et al. (eds.), *Protocol Test Systems*

and LOTOS can describe partially-specified behavior. Therefore, the state machines abstracted from the specifications may be *both* partially-specified *and* nondeterministic. There is a practical need for testing nondeterministic models [49]; in particular, communication protocols, when tested under the ISO remote testing architecture, are often modeled as partially-specified and nondeterministic finite state machines. Vuong, Loureiro and Chanson discussed how and why nondeterminism is introduced into communication protocols [48].

Some work on test generation for nondeterministic models has been done in the context of LOTOS [45, 36, 6] and finite labeled transition systems [15, 16], but they are not applicable to testing nondeterministic state machines where every transition is associated with an input/output pair. Furthermore, several results have been reported on test generation for either partially-specified deterministic machines [33, 12, 13], or completely-specified nondeterministic machines [26, 23, 29]. The methods given in [26, 23] are all based on the generalization of unique I/O sequences [39], even when applied to FSM's, a specific class of NFSM's, they still cannot guarantee full fault coverage, although full fault coverage for FSM's can be assured by many other methods if the number of states in any implementation is bounded by a kown integer. The reason is the same as pointed out in [47]. (Note: We asume in this paper that the number of states in any implementation is bounded by a known integer. Without this assumption no method can guarantee full fault coverage.) Therefore, they have limited fault detection power. A generalized Wp-method has been proposed in [29], which can guarantee full fault coverage under the assumption that the number of states in any implementation is bounded. Initial work on test generation for *both* partially-specified *and* nondeterministic finite machines has been given in a department report [28], which is also presented in this paper.

We study in this paper test generation for the finite state machines that could be *both* partially-specified *and* nondeterministic, guided by pre-defined conformance relations. For completely-specified deterministic finite state machines (FSM's), partially-specified deterministic finite state machines (PFSM's), and completely-specified nondeterministic finite state machines (NFSM's), there are commonly-defined conformance relations in the literature [14, 10, 33, 35, 46, 43, 17]. A conformance relation for partially-specified nondeterministic finite state machines (PNFSM's) was first presented in [28], which was also explained in [35].

In Section 2, after formally defining PNFSM's and several related notations, we present a conformance relation, called *quasi-equivalence*, for PNFSM's [28]. The relation is defined in terms of input/output traces in accordance with black-box testing strategy. When the relation is applied to FSM's, NFSM's and PFSM's, which are typical classes of PNFSM's, it coincides to corresponding conformance relations given in the literature. We also define several concepts which are related to testing.

Guided by the conformance relation, in Section 3, we come out with a method for generating test cases from PNFSM's. We first transform a PNFSM to an equivalent one that has a lower degree of nondeterminism, called *observable* PNFSM (OPNFSM). The OPNFSM's have the property that a state and an input/output pair uniquely determine the next state, while a state and an input alone do not necessarily determine a unique next state and an output. We then generate test suites from the resulting OPNFSM by a method which we call *Harmonized State Identification method* (HSI-method). As an example, we applied in [28] the method to generate a test suite for a communication protocol, called *Inres*, within the remote testing architecture.

In Section 4, we compare our method with other test generation methods, on the basis of applicability, fault coverage and the size of test suites. The main advantage of our method over the other methods is its broadest applicability with guaranteed fault coverage for prescribed faults. We give concluding remarks in Section 5.

2. NOTATIONS AND ABSTRACT TESTING FRAMEWORK

We first give in this section the definition of PNFSM's, then present conformance relations for PNFSM's under the black-box testing strategy (where implementations are assumed to be black-boxes), and finally define several other concepts related to testing.

2.1 Partially-specified nondeterministic finite state machines (PNFSM's)

We first define PNFSM's in a traditional form similar to that given in [43] for NFSM's. For the convenience of presentation, we then introduce additional notations for PNFSM's similar to those for labeled transition systems [6, 15, 16]; we also define several specific classes of PNFSM's.

A *Partially-specified Nondeterministic Finite State Machine* (PNFSM) is defined as a 5-tuple (St, Li, Lo, h, S_0) where:

1) St is a finite set of states, St=$\{S_0, S_1, ..., S_{n-1}\}$.
2) Li is a finite set of inputs.
3) Lo is a finite set of outputs.
4) h is a behavior function:

h: $\mathcal{D} \Rightarrow$ powerset(St × Lo) $\setminus\{\varnothing\}$ where

(1) $\mathcal{D} \subseteq$ St × Li (PNFSM becomes completely specified if $\mathcal{D}$=St × Li);

(2) $\varnothing$ denotes the empty set;

(3) Given two sets A and B, $A \setminus B = \{x \mid x \in A \ \& \ x \notin B\}$.

Let P, Q∈ St, a∈ Li and b∈ Lo. We write P-a/b->Q to denote (Q, b)∈ h(P,a); P-a/b->Q is called a *transition* from P to Q with label a /b.

5) S_0 is the initial state, which is in St. □

We assume that a "reliable" reset input *r* is available in any implementation of a PNFSM such that upon receiving *r* in any state the implementation returns to the initial state. We do not include spontaneous transitions (or called internal actions) in our model, since the PNFSM's that allow spontaneous transitions can be modeled by equivalent PNFSM's without spontaneous transitions using an approach similar to the one in [27].

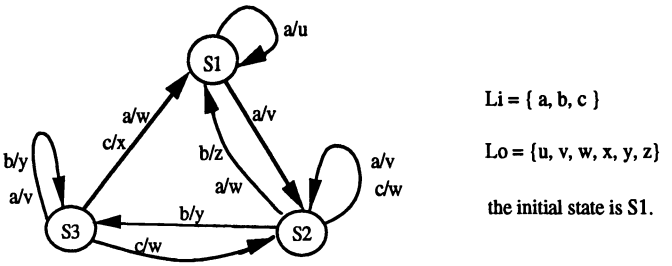


Figure 1. An example of a partial NFSM

We often use in the following the term "partial machine" to refer to a PNFSM, which may be deterministic or not. A partial machine can be represented by a directed graph in which the nodes are the states and the directed edges are transitions linking the states. Figure 1 shows an example of such a machine. For a PNFSM, if no two outgoing transitions from the same state have the same input, then the machine is deterministic; and we call it a partial FSM (PFSM). For the convenience of the presentation, we also introduce in Table 1 several notations.

Table 1. Notation for PNFSM's

notation	meaning
L	Li × Lo, a set of input/output pairs; u denotes such a pair
ε	ε is the empty sequence.
L*	set of sequences over L; x denotes such a sequence. Note that ε∈ L*
P\-u->	For P, Q ∈ St, not(∃Q(P-u->Q))
P=ε=>Q	P=Q
P=a/b=>Q	P-a/b->Q
P=x=>Q	$\exists P_1, ..., P_{k-1} \in St$ ($P=P_0=u_1=>P_1...=u_k=>P_k=Q$) where $u_1, ...,u_k \in L$, and $x=u_1...u_k$
P=x=>	∃Q∈ St (P=x=>Q)
Tr(P)	*Tr*(P)={ x | P=x=>}
x^{in}	For x∈ L*, x^{in} is an input sequence obtained by deleting all outputs in x (note that $x^{in} \in Li^*$)
V^{in}	For V⊆L*, $V^{in}=\{x^{in} \mid x \in V\}$
Tr^{in}(P)	Tr^{in}(P)={ x^{in} | P=x=> }, (note that Tr^{in}(P)=Li* for each state P of completely-specified NFSM's)

Given an PNFSM S, we say that S is *initially connected* if every state is reachable from the initial state by a directed path in the machine; i.e., $\forall S_i \in St\ \exists x \in L^*$ ($S_0=x=>S_i$). Without loss of generality, we assume that all PNFSM's considered in the rest of the paper are initially connected. If a given PNFSM S is not initially connected, we may consider only such a submachine which is a portion of S consisting of all states and transitions that are reachable from the initial state of S. The unreachable states and transitions of machines do not affect the behavior of the machines.

We now define several specific classes of PNFSM's, which are useful concepts for test generation. We first define so-called observable PNFSM's (OPNFSM's), a concept originally described in [43] for completely specified machines, which is a specific class of PNFSM's. A PNFSM is said to be *observable* if for every state S∈ St, and every input/output pair a/b ∈ L, there is at most one transition from S with the same pair. Formally, a PNFSM is observable if $\forall S \in St\ \forall u \in L$ (S-u->S_i & S-u->$S_j \Longrightarrow i=j$). As an example, the PNFSM shown in Figure 1 is an OPNFSM. OPNFSM's are a subclass of partial machines. In observable machines, a state and an input/output pair can uniquely determine at most one next state. However, an OPNFSM may still be nondeterministic in the sense that a state and an input cannot determine a unique next state and a unique output. We note that all deterministic machines are observable.

A PNFSM is said to be *reduced* if none of its states accept the same set of input/output sequences. Formally, a PNFSM is reduced if $\forall S_i, S_j \in St$ ($i \neq j \Longrightarrow Tr(S_i) \neq Tr(S_j)$). Given a PNFSM, two states are said to be *distinguishable* if there is an input/output sequence x such that x can be accepted by only one of the two states but the input sequence x^{in} can be accepted by both of them. Formally, given a pair of states S_i and S_j, S_i and S_j are said to be *distinguishable*, written $S_i \not\cong S_j$, if $\exists x \in Tr(S_i) \oplus Tr(S_j)$ ($x^{in} \in Tr^{in}(S_i) \cap Tr^{in}(S_j)$ where $Tr(S_i) \oplus Tr(S_j) = (Tr(S_i) \cup Tr(S_j)) \setminus (Tr(S_i) \cap Tr(S_j))$. If a pair of states are not distinguishable, we say that they are *indistinguishable*. For example, the states S0 and S1 of the machine shown in Figure 1 are distinguishable; and the states S1 and S3 are indistinguishable.

A PNFSM is said to be *minimal* if every pair of different states are distinguishable. Formally, a PNFSM is minimal if $\forall S_i, S_j \in St$ ($i \neq j \Longrightarrow S_i \not\equiv S_j$). A minimal PNFSM is reduced, but a reduced PNFSM is not necessarily minimal. Given a minimal machine S, each state is distinguishable from all other states; however, this is not necessarily true for a reduced machine. If we consider a completely specified machine, then a reduced machine is also minimal. The OPNFSM shown in Figure 1 is reduced, but not minimal.

2.2. Conformance relations for PNFSM's

Before any study on how to generate test suites for PNFSM's, the following question must first be answered: under the black-box testing strategy, what kind of conformance relation between a specification and the corresponding implementation is expected to hold ? There are several conformance relations defined in the literature for finite state machines and process algebras. Generalizing the conformance relations for finite state machines (e.g. FSM's, PFSM's and NFSM's) on the basis of intuitive notions, we will define in this section conformance relations for PNFSM's in terms of the relations between their initial states.

For (completely-specified, deterministic) FSM's, there is a widely-accepted conformance relation, called *equivalence*, (see, e.g., [14, 10, 46, 43, 17]), which requires that a specification and its implementation produce the same output sequence for every input sequence.

The *equivalence* relation between two states P and Q in PNFSM's, written $P \equiv Q$, holds if $Tr(P) = Tr(Q)$. Given two PNFSM's S and I with their initial states S_0 and I_0, we write $S \equiv I$ if $S_0 \equiv I_0$. □

We say that an implementation I is *equivalent* to its specification S if and only if $S \equiv I$. The above definition is similar to that in [14, 10, 46, 17], but it can also be applied to PNFSM's. The above relation is an equivalence relation since it is reflective, transitive and symmetric. It corresponds to the equivalence relation between NFSM's given in [43]. However, this definition ignores the fact that after a given partial trace, the machine may be in one of several different states, and the inputs for which transitions are defined may be different for these different states. In the context of process algebras, these considerations lead to the consideration of blocking, also called failures [19]. In order to avoid these complications, we limit ourselves in the following to the consideration of observable PNFSM's mainly, for which the past input/output trace determines the state of the machine.

We now explain the intuitive notions for defining a conformance relation for partial machines. We say that a state machine is partial if its behavior function is not defined for all state/input combinations. There may be different reasons for introducing partial machines. There are (at least) three basic interpretations for an undefined state/input combination, namely "blocking", "don't care" and "forbidden" [35]. The blocking interpretation is usually adopted with process algebra approaches and means that the machine should not accept such an input. In this paper we assume, on the contrary, that the machine cannot be blocked by an input from the environment.

In the case of the "don't care" interpretation, an undefined state/input combination means that the specification allows any further behavior of an implementation starting from the given state under the given input. Since an implementation can always be represented by a completely specified machine it actually completes a given partially specified machine. In other words, a partial machine represents a set of completely specified machines, and its implementation is required to conform to one of these machines.

In the last interpretation, an undefined state/input combination means that the input in the combination cannot be applied to the state, i.e., a transition cannot be executed, due to limitations imposed by the environment. For example, it is impossible to send data to a protocol machine via a connection until it has accepted this connection. Undefined "forbidden" state/input combinations will never occur in real executions. Thus, any method for executable test suite derivation should not consider these combinations.

These interpretations require that the external behavior of an implementation is equal to that of its specification only for all those input sequences that can be accepted by a specification. For PFSM's (a specific class of PNFSM's), a conformance relation, called *quasi-equivalence*, was presented in [33, 43, 17], which is in accordance with the above intuitive notions. The relation requires that, *for every input sequence that can be accepted by a specification, the specification and its implementation produce the same output sequence.*

Guided by the same intuitive notions, we generalize the quasi-equivalence to PNFSM's by requiring that, *for every input sequence that can be accepted by a specification, the specification and its implementation produce the same set of output sequences.* We formally define the generalized quasi-equivalence as follows.

DEFINITION *Quasi-equivalence*:
The *quasi-equivalence* relation between two states P and Q in PNFSM's, written

$P \leq_{quasi} Q$, holds iff

(a) $Tr(P) \subseteq Tr(Q)$, and

(b) $\forall x \in Tr(Q)\ (\ x^{in} \in Tr^{in}(P) \implies x \in Tr(P)\ )$

Given two PNFSM's S and I with their initial states S_0 and I_0, we write $S \leq_{quasi} I$ (i.e., implementation I is *quasi-equivalent to* its specification) iff $S_0 \leq_{quasi} I_0$. □

We present in the following the relationship between the above-defined conformance relations.

THEOREM 1: Given two PNFSM's S and I, assuming that they have common Li and Lo, we have the following statements:

(i) $S \equiv I \iff S \leq_{quasi} I \ \&\ I \leq_{quasi} S$

(ii) if S and I are (completely-specified) NFSM's, then $S \leq_{quasi} I \iff S \equiv I$. □

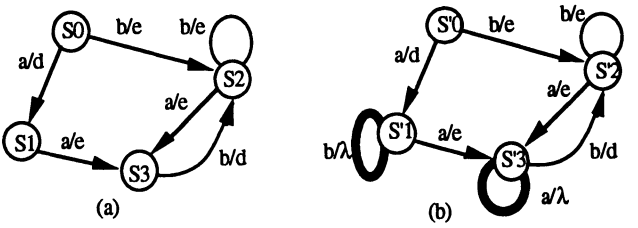
The above theorem is evident from the corresponding definitions.

It is well-known that any nondeterministic finite automaton where each transition is associated with a single symbol (not with an I/O pair) can be modeled by an equivalent deterministic automaton [20]. However, nondeterministic finite state machines, where each transition is associated with an I/O pair, cannot be modeled by equivalent deterministic finite state machines. For example, in a NFSM with S_0-a/b-> and S_0-a/c->, we have {a/b, a/c} $\subseteq Tr(S_0)$. On the other hand, no deterministic FSM has {a/b, a/c} $\subseteq Tr(S_0)$. Therefore, nondeterministic finite state machines, in general, cannot be transformed to equivalent deterministic finite state machines for test generation.

2.3. Necessity of specific treatment for partially-specified machines

We further discuss in this section why the test generation methods for completely-specified machines are not adequate for partially-specified machines.

In the literature, the so-called completeness assumption was proposed to handle the test generation for partially-specified machines [39]. It assumes that a PFSM actually represents an (completely-specified) FSM resulting from the PFSM in the following manner: for every undefined state/input combination in the PFSM, add a self-loop transition to the state with the input and a null output λ. For example, the PFSM S shown in Figure 2a represents the FSM shown in Figure 2b. Using this assumption, PFSM's are only used as a shorthand notation for their implied FSM's; and the test generation methods for such PFSM's are just the one for FSM's.



Note: Li = { a, b }, Lo = { d, e, λ }. The initial state is S0, and λ stands for a null output.
Figure 2. (a) A specification PFSM S, (b) the implied FSM S' of S.

PFSM's could also be a shorthand notation for the FSM's derived in the following manner [35]: for every undefined state/input combination in the PFSM's, add a transition to an "error" (terminal) state with an "error" output. In such a case, there is no need to handle the partially specified machines.

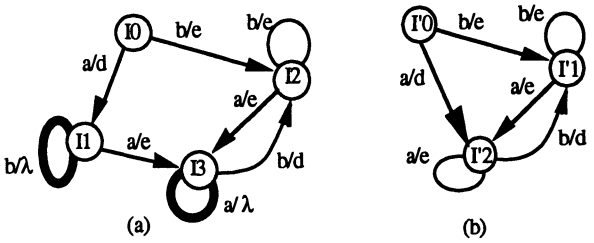


Figure 3. (a) an implementation FSM I, (b) an implementation FSM I'.

However, PFSM's are not necessarily used only as a shorthand notation for FSM's. As discussed before, an undefined state/input combination in a PFSM could represent the "don't care" situation (for example, in the state machines abstracted from the ISO transport protocol) [35]. In this case, a PFSM is not a shorthand notation for a single FSM. As an example, consider the PFSM S shown in Figure 2a, and the FSM's I and I' shown in Figures 3a and 3b. According to the "don't care" interpretation, both I and I' are valid implementations of S. In this case, if the completeness assumption is followed and if the test sequences are generated from the implied FSM using the existing methods for FSM's, the valid implementation I' could be determined as an invalid implementation when such test sequences are used since I' is not equivalent to the implied FSM S' of S. Therefore, the test generation methods for FSM's and NFSM's are not applicable to PNFSM's, and we need to develop test generation methods for PNFSM's. Further discussion can be found in [35].

We note that the quasi-equivalence captures the intutive notion of "don't care" interpretation, as discussed in a former section. The above I' will be considered as a valid implementation if a test generation method with respect to this relation is used. This relation serves as a basis for test generation for PNFSM's. Therefore, we need to study the test generation for PNFSM's with respect to this relation.

2.4. Definitions related to testing

We define in this section several concepts which are related to testing nondeterministic finite state machines.

For a given PNFSM, a sequence t of a finite length is said to be a ***test sequence*** if $t \in Tr^{in}(S_0)$. A ***test suite*** is a finite set of test sequences.

We need a concept called *equivalence with respect to a given input set* Π ($\Pi \subseteq Li^*$) for explanation of validity of our test generation method. The equivalence relation with respect to a given input set Π requires that, *for every input sequence in* Π *that can be accepted by both a specification and its implementation, the specification and its implementation produce the same set of possible output sequences,* which is formalized as follows.

DEFINITION: *Equivalence with respect to a given input set.*

The *equivalence* relation between two states P and Q, with respect to a given input set $\Pi \subseteq Li^*$, written $P =_{\Pi} Q$, holds if $V \cap Tr(P) = V \cap Tr(Q)$

where $V = \{x \mid x \in L^* \ \& \ x^{in} \in \Pi \cap Tr^{in}(P) \cap Tr^{in}(Q)\}$.

Given two PNFSM's S and I with their initial states S_0 and I_0, we write $S =_{\Pi} I$ if $S_0 =_{\Pi} I_0$.

Furthermore, we say that P is Π-*equivalent to* Q if $P =_{\Pi} Q$. □

The relation is reflective and symmetric but not transitive.

We also need the following notations for presenting our test selection method. Given a set of sequences $V \subseteq Li^*$, *prefix set* *pref*(V) *for a given set of sequences* is defined as

$pref(V) = \{t1 \mid t2 \in Li^* \ \& \ t1.t2 \in V \ \& \ t1 \neq \varepsilon\}$ where t1.t2 is the concatenation of t1 with t2.

For example, let V={ a.b, a.c.d}, then *pref*(V)={a, a.b, a.c, a.c.d }.

Given sets $V1, V2 \subseteq L^*$ (or $V1, V2 \subseteq Li^*$), *the concatenation of sets of sequences*, written ".", is defined as follows: $V1.V2 = \{ t1.t2 \mid t1 \in V1 \ \& \ t2 \in V2 \}$ where t1.t2 is the concatenation of t1 with t2. We write $V^n = V.V^{n-1}$ for $n > 1$ and $V^1 = V$.

In order to test nondeterministic implementations, one usually makes a so-called *complete-testing assumption*: it is possible, by applying a given input sequence to a given implementation a finite number of times, to exercise all possible execution paths of the implementation which are traversed by the input sequence [15, 16, 29]. Without such an assumption, no test suites can guarantee full fault coverage (in terms of conformance relations) for nondeterministic implementations. This assumption is similar to the one of so-called "all weather conditions" for nondeterministic systems of Milner [31, page 11]. For testing nondeterministic models, as pointed in [6], the quality of testing increases with the number of repetition of test sequence application; in actual testing, this number is limited by practical and economical considerations. Ideally, for an implementation and a given input sequence, the probability that not all possible corresponding execution paths are exercised at least once, can usually be reduced to close to zero by applying the input sequence a sufficiently large number of times. We assume in the rest of the paper that this assumption holds for implementations under discussion unless we specify the other case explicitly.

3. TEST GENERATION

We present in this section a test generation method for PNFSM's, which we call *Harmonized State Identification method* (HSI-method). The test suites generated by the HSI-method can be used to test implementations against their PNFSM specifications with respect to the quasi-equivalence relation.

3.1. Test generation for OPNFSM's

We first define several key concepts for presenting our method, then give an algorithm of generating test suites, and finally present a theorem for establishing the validity of the algorithm.

DEFINITION: *Characterization set* W:

Given an OPNFSM, a *characterization set* is a minimal set $W \subseteq Li^*$ such that:

$\forall S_i, S_j \in St \ (S_i \not\simeq S_j \implies \exists x \in Tr(S_i) \oplus Tr(S_j) \ (x^{in} \in Tr^{in}(S_i) \cap Tr^{in}(S_j) \cap W))$. □

The above definition is generalized from the concept of the characterization set for FSM's given in [10] to PNFSM's. The W set is used to identify states in a given machine. Intuitively, for any two states that are distinguishable, there exists an input sequence in W such that (i) it is acceptable in the two states and (ii) two different sets of output sequences are produced when this input sequence is applied to these states, respectively. An algorithm of generating characterization sets is given in [28].

We find, however, that it is not neccesary to use the whole characterization set for state identification. We only use the subsets of this set, called *harmonized state identification sets*, for state identification.

DEFINITION: *Harmonized state identification sets* $\{D_0, D_1, ..., D_{n-1}\}$:

Given an OPNFSM with n states, $\{D_0, D_1, ..., D_{n-1}\}$ is a tuple of *harmonized state identification sets* if, for i=0, 1, ..., n-1, D_i is a minimal set such that

(i) $D_i \subseteq Tr^{in}(S_i) \cap pref(W)$, and

(ii) for j=0, 1, ..., n-1, $S_i \not\equiv S_j \Longrightarrow \exists x \in Tr(S_i) \oplus Tr(S_j)$ ($x^{in} \in pref(D_i) \cap pref(D_j)$). □

Intuitively, for any two states that are distinguishable, there exists an input sequence in $pref(D_i) \cap pref(D_j)$ such that two different sets of output sequences are produced when this input sequence is applied to these states, respectively. For the OPNFSM shown in Figure 1, $D_1=D_2=D_3=\{a.b\}$. An algorithm of generating harmonized state identification sets is given in [28].

DEFINITION: *subscripts*(A) for a given state set A:

For $A \subseteq St$, *subscripts*(A) is the sequence of integers i1, i2, ..., ik such that

$$i1 < i2 < ... < ik \text{ and } A=\{S_{i1}, S_{i2}, ..., S_{ik}\}.$$ □

Given two sets of states A and B, we say that the *subscripts* of A is *smaller* than that of B if *subscripts*(A) precedes *subscripts*(B) in lexicographic order. The notation *subscripts*(A) for a given set of states A is needed for defining a so-called *maximal set of pairwise-distinguishable states* $f(S_i)$ *for a given state* S_i for OPNFSM's. The states in a given set are pairwise-distinguishable if and only if every pair of states in the set are distinguishable. A maximal set of pairwise-distinguishable states is a set such that it is not contained in any other set of pairwise-distinguishable states. A maximal set of pairwise-distinguishable states $f(S_i)$ for a given state S_i is the set with the smallest subscript among the maximal sets of pairwise-distinguishable states that contains S_i, which is formally defined as follows.

DEFINITION: *Maximal set of pairwise-distinguishable states* $f(S_i)$ *for a given state* S_i:

Given an OPNFSM and a state $S_i \in St$, $f(S_i)$ is defined as a set $A \subseteq St$ such that:

(i) $S_i \in A$, and

(ii) $\forall S_k, S_j \in A$ ($k \neq j \Longrightarrow S_k \not\equiv S_j$), and

(iii) there is no $B \subseteq St$ such that

(i') $S_i \in B$, and

(ii') $\forall S_k, S_j \in B$ ($k \neq j \Longrightarrow S_k \not\equiv S_j$), and

(iii') $|B| > |A|$ or $|B|=|A|$, and *subscripts*(B) precedes *subscripts*(A) in lexicographic order. □

Given a minimal machine, for every state S_i, we have $f(S_i)=St$. For a given OPNFSM, we denote the number of all different maximal sets of pairwise-distinguishable states as *fuzziness degree* δ, as defined below.

DEFINITION: *Fuzziness degree* δ *for a given* OPNFSM:

Given an OPNFSM, we have $\delta = |\{f(S_i) \mid S_i \in St\}|$. □

According to the above definition, every state S_i has only one maximal set of pairwise-distinguishable states $f(S_i)$. Therefore, it is easy to see that $1 \leq \delta \leq |St|$, and $\delta=1$ for any minimal

OPNFSM. A fuzziness degree δ of a given OPNFSM influences the size of the test suite and the lengths of the test cases.

In order to present our method, we need a concept, called *prime machine*. For a given PNFSM S (St, Li, Lo, h_S, S_0), the *prime machine* of S is defined as a reduced (not necessarily minimal) OPNFSM M (St_M, Li, Lo, h_M, M_0) such that S≡M. In test generation, the role of prime machines for PNFSM's is the same as that of minimal FSM's for (deterministic) FSM's. For two given PNFSM's S1 and S2, if S1≡S2, then their prime machines, say M1 and M2, are isomorphic to one another. It is easy to prove that the equivalence relation is the isomorphism between M1 and M2. The isomorphism of various state machines were studied in [10] and [43].

We note: All existing test generation methods for (deterministic) FSM's that ensure full fault coverage (for examples, the W- and Wp- methods [10, 14]), assume an *upper bound* on the number of states in the prime machine of the given FSM implementation. Similarly, our test generation method also assumes an *upper bound* on the number of states in the prime machine of the given NFSM implementation. In the simplest case, one may assume that this number is equal to the number of states of the specification. Without the assumption of such an upper bound, no method can check the equivalence between specifications and their implementations for FSM's, as pointed out in [17]. Therefore, this assumption is also necessary for PNFSM's. We also note: although specifications are considered to be partially specified, implementations are usually completely specified. Therefore, we assume in this paper that implementations are (completely specified) NFSM's. We give in the following the test generation algorithm, which we call *Harmonized State Identification method* (HSI-method). This algorithm also assumes that the number of states in the prime machine of the given NFSM implementation is bounded by an integer m.

ALGORITHM 1: Test generation.

Input : A specification S in the form of an (arbitrary) OPNFSM (St, Li, Lo, h, S_0), and the upper bound m on the number of states in the prime machine of the given NFSM implementation.

Output : A test suite $\prod$.

Step 1: Determine the fuzziness degree δ of S.

Step 2: Let the number of states in S be n ($n \le \delta m$). Find a set of harmonized state identification sets $\{D_0, D_1, ..., D_{n-1}\}$ for S.

Step 3: Construct a minimal set of input sequence $\mathbb{Q} \subseteq Li^*$ such that:

$$\forall S_i \in St \ \exists x \in L^* \ (x^{in} \in \mathbb{Q} \ \& \ S_0 = x => S_i).$$

Step 4: Construct a test suite $\prod$ such that:

$$\prod = \bigcup_{\substack{S_0 = x => Si \ \& \\ x^{in} \in Q.(\{\varepsilon\} \cup Li ... \cup Li^{\delta m - n + 1})}} \{x^{in}\}.D_i$$

□.

We note that this algorithm has some similarity with test selection using the W-method [10], however it produces smaller test suites than the W-method when applied to the FSM's. The reason is that D_i is smaller than the W set for any i. In Algorithm 1, the given specification is not required to be reduced or minimal. However, a much smaller test suite will be obtained if we use its reduced form.

As an example, we derive a test suite $\prod$ for the PNFSM given in Figure 1 as follows:

$$\mathbb{Q} = \{\varepsilon, a, a.b\}, \quad D_1 = D_2 = D_3 = \{a.b\}, \quad f(S_1) = \{S_1, S_3\}, \quad f(S_2) = f(S_3) = \{S_2, S_3\}, \quad \delta = 2.$$

Assume that the prime machines of implementations do not have more than 2 states (i.e., m=2); then, we have $n \leq \delta m$. We note that a test suite could be reduced by deleting each test case that is a prefix of another test case. The final test suite is as follows:

Π = {a.a.a.b, a.a.b.a.b, a.a.c.a.b, a.b.a.a.b, a.c.a.a.b, a.c.b.a.b, a.c.c.a.b, a.b.a.a.a.b, a.b.a.b.a.b, a.b.a.c.a.b, a.b.b.a.a.b, a.b.b.b.a.b, a.b.b.c.a.b, a.b.c.a.a.b, a.b.c.b.a.b, a.b.c.c.a.b, a.c.a.b, a.b.b.a.b, a.b.c.a.b, a.a.a.a.b}

We note that a reset must be issued before the execution of each test case.

The validity of the HSI-method is intuitively explained as follows. Consider a given specification S in the form of an OPNFSM, and any NFSM I; let M be the prime machine of I. Assume that the number of states in M is bounded by m. Assume $S=\Pi I$. Then,

(1) We say that a subset of Li* is a *state cover* if, for every state S_i, the subset contains an input sequence that may lead the machine from the initial state S_0 to S_i.
The set $\mathbb{Q}$ represents a state cover for the specification S.

(2) We say that a subset of Li* is a *transition cover* if, for every transition S_i-u->S_j, the subset contains an input sequence $x^{in}.u^{in}$ such that x^{in} and $x^{in}.u^{in}$ may lead the machine from the initial state S_0 to S_i and S_j, respectively.
The set $\mathbb{Q}.(\{\varepsilon\} \cup Li \cup ... \cup Li^{\delta m-n+1})$ represents a transition cover for the prime machine M of the implementation. Therefore, in Step 4 of the above algorithm, $\{x^{in}\}.D_i$ is used to check each transition in implementation. Furthermore, in the definition of harmonized state identification sets, we required that there exists an input sequence in $pref(D_i) \cap pref(D_j)$ such that two different sets of output sequences are produced when this input sequence is applied to these states, respectively. This requirement enables us to verify every state in implementation also.

(3) In Step 4 of the above algorithm, the formula "$\delta m-n+1$" plays the key role for this construction, similar to the "m-n+1" of the test suites of the Wp-method [14] for the transition cover. The additional δ is caused by indistinguishable states in PNFSM's. The indistinguishable states in a reduced partialy specified machine specification make test generation more difficult that one for completely specified machines. For two indistinguishable states P, Q in such specification, we may have a single state M in its implementation such that both $P \leq_{quasi} M$ and $Q \leq_{quasi} M$ hold; furthermore, a single state P in a reduced partialy specified machine specification can be represented (implemented) by several states M_1, M_2, ..., in its implementation (i.e., $P \leq_{quasi} M_1$, $P \leq_{quasi} M_2$, ...) (note: if the specification is a completely specified minimal machine, instead of the above, we have one-to-one correspondence). We need to test such different matching. In order to do so, the test suite is influenced by the formula $\delta m-n+1$, in which the fuziness degree δ is a kind of measure for indistinguishable states. The more indistinguishable states the specification has, the greater δ is; thus the size of the test suite is greater. In our experience, although the PNFSM's obtained from communication protocols are often not completely specified (still partial machines), yet the fuziness degree δ's of these machines are often equal to one.

The validity of the test generation method is formally presented and proved in the following theorem.

THEOREM 2: (Validity of the test generation method):
Consider a given specification S in the form of an OPNFSM, and any NFSM I. Suppose $n \leq \delta m$ where n is the number of states in S, and m is the upper bound on the number of states in the prime machine of I. Let Π be the test suite generated for S using Algorithm 1. Then, $S \leq_{quasi} I$ iff $S=\Pi I$. The proof is given in [28]. □

As shown in Algorithm 1, test suites for minimal partial machines can be constructed in the same way as for completely specified minimal machines since δ is equal to one for minimal machines. However, if a partial machine has indistinguishable states, then the machine cannot be transformed into its minimal form to generate a test suite with respect to the quasi-equivalence relation. The reason is that the transformation of a partial machine into a minimal form, through merging states, will result in the appearance of new traces that are not defined in the original machine. In turn, this implies that some valid implementations may not pass a test suite derived from the minimal form, and that some test cases in such a test suite may not be acceptable in the original machine. Therefore, partial machines should not be transformed into minimal forms for test generation.

In practical applications, state machines that represent implementations are always completely specified. Therefore, for a given OPNFSM specification S and a given test suite Π, if the complete-testing assumption is satisfied by a given implementation NFSM I, for every test case t in Π, then the relations "$S=_{\Pi}I$" can be checked as follows. For every test case t in Π, let A and B be the two sets of output sequences produced when t is applied to S and I, respectively. A is easy to be obtained from S. B can be derived through appying the test case t a sufficient number of times if the complete-testing assumption is satisfied by I. If for every test case t in Π, A=B, then "$S=_{\Pi}I$". Thus, according to Theorem 2, the test suites generated by Algorithm 1 can be used to test NFSM implementations against their specification with respect to the quasi-equivalence relation.

3.2. Equivalent transformation to obtain OPNFSM's

We now discuss the method of transforming arbitrary PNFSM's into trace-equivalent OPNFSM's. Combined with this transformation, the HSI-method can be used to generate test sequences for arbitrary PNFSM's. As discussed in Section 2.2, in general, NFSM's cannot be transformed into trace-equivalent DFSM's, but they can be transformed into trace-equivalent ONFSM's. The method of transforming nondeterministic finite automata into equivalent deterministic automata given in [20] can be modified to transform arbitrary NFSM's into trace-equivalent ONFSM's. The main idea of the modification is as follows: (1) Given an NFSM F1, by viewing input/output pairs in F1 as labels, consider the NFSM F1 as a nondeterministic finite automaton A1. (2) Apply the method given in [20] to A1 to obtain an equivalent deterministic finite automaton A2. (3) By viewing the labels in A2 as original input/output pairs, derive an ONFSM F2 from A2. The details of a transformation algorithm is samilar to the methods given in [28, 29].

4. COMPARISON WITH OTHER RELATED WORK

4.1. Comparison with the existing test generation methods for I/O state machines

Former work on test generation for I/O state machines mainly concentrated on FSM's, NFSM's and PFSM's. No work on test generation for PNFSM's has been reported with respect to the quasi-equivalence relation. We now solved this problem since our HSI-method is applicable to PNFSM's. The applicability of our HSI-method is better than that of related existing methods. Since FSM's, NFSM's and PFSM's are specific classes of PNFSM's, the HSI-method is applicable to each of them, for testing the traditional equivalence and quasi-equivalence relations which are handled by existing methods [14, 39, 10, 13, 33, 12, 29], respectively (see Theorem 1).

We now compare our method with the existing methods which also provide full fault coverage. We first compare our method with the W- and Wp- methods [10, 14], which are applicable only to FSM's. When the HSI-method is applied to FSM's, it can produce smaller test suite than the W-method since each harmonized state identification set is usually smaller (never greater) than the characterization set. However, neither the HSI-method nor the Wp-

method necessarily produces smaller test suites than the other, since each harmonized state identification set is even usually smaller (never greater) than the characterization set but it could be greater than the W_i set of the Wp-method. When compared with the generalized Wp-method method given in [29], which is only applicable to NFSM's (incuding FSM's), for the reason similar to the above, neither the HSI-method nor the generalized Wp-method necessarily produces smaller test suites than the other. We also note that, compared with the method given in [13, 33], which is also based on haromonized state identification sets and fuzziness degree, but is applicable only to PFSM's, neither our method nor that method necessarily produces smaller test suites than the other. More detailed comparison can be found in [28]. In conclusion, the applicability of the HSI-method is better than that of related existing methods which have been reported; i.e., our HSI-method is applicable to PNFSM's. None of existing work on testing PNFSM's has been reported before with respect to the quasi-equivalence relation. Furthermore, the HSI-method will not produce greater test suites than the existing methods for FSM's, NFSM's and PFSM's.

4.2. Comparison with the work on test generation for labeled transition systems

One of important issues in protocol conformance testing is to test communication protocols specified in LOTOS [3, 5]. Fundamental framework of protocol conformance testing of LOTOS has been initially given in [5, 6], in terms of the *labeled transition systems* (LTSs). In particular, a so-called **conf** relation and *canonical tester* become a basis for a large body work [36, 5, 6, 7, 44]. The **conf** relation was developed to describe the relationship between a specification and the corresponding valid implementations; and the canonical tester is viewed as a test suite for testing implementations against their LOTOS specifications with respect to **conf**. Brinksma et al also suggested the so-called *failure preorder* as implementation relation [7], which belongs to the so-called *robust testing* defined in [5]. Drira et al suggested some improvement for the **conf** [11], also involving in the robust testing.

However, for checking **conf** relation, existing test generation methods produce the canonical tester [36, 5, 6, 44], which represent an infinite set of test cases if LTS specifications have directed loops. For practical applications, test suites should be finite. Our test generation method for PNFSM's produces a finite set of test suite.

The test generation works on I/O state machines and on LTSs used to be investigated separately. Certain initial effort has been done to bridge these two areas by showing how to model LTSs with I/O state machines [35]; however, the relationship between the various conformance relations of I/O state machines and LTSs needs further investigation (note: In the area of LTSs, the term "implementation relations" was used to refer to the concept of conformance relations). It is believed that, using different methods to model LTSs in I/O state machines, the quasi-equivalence relation could correspond to different conformance relartions (i.e. implementation relations) of LTSs [35]. This implies that our method could be useful for LTSs, in the area of testing LOTOS specified protocols, to check different conformance relartions of LTSs.

5. CONCLUDING REMARKS

We first propose in this paper a conformance relation, called *quasi-equivalence*, as a framework for testing arbitrary partially-specified, even nonminimal and nondeterministic finite state machines (PNFSM's). This relation is a generalized version of several other conformance relations for FSM's, NFSM's, and PFSM's. We then present a uniform method, called the HSI-method, for generating test suites for different types of state machines, ranging from pure FSM's to arbitrary partially-specified, even nonminimal, nondeterministic finite state machines. This method can be applied to test generation for the control part of specifications written in SDL or ESTELLE. In such cases, we can first abstract SDL processes or ESTELLE modules to PNFSM's by neglecting parameters [25, 30]; we then apply the test generation method for

the resulting PNFSM's. In the situation of testing concurrent programs specified in SDL or ESTELLE, even though individual processes are deterministic, the whole system usually is nondeterministic; therefore, there is a need for methods to test nondeterministic machines. As far as implementation of test generation tools is concerned, the advantage of our method is that we need to implement only one test generation method -- the HSI-method -- for PNFSM's, instead of implementing several individual methods for FSM's, PFSM's and NFSM's since they are specific cases of partially-specified nondeterministic finite state machines.

REFERENCES

[1] Alfred V. Aho, Barry S. Bosik and Stephen J.Griesmer, "Protocol Testing and Verification within AT&T", AT&T Technical Journal, Vol.69, No.1, 1990, pp.4-6.

[2] AT&T Technical Journal, Special Issue on Protocol Testing and Verification, Vol.69, No.1, 1990.

[3] T. Bolognesi, Ed Brinksma, "Introduction to the ISO Specification Language LOTOS", Computer Networks and ISDN Systems, Vol.14 (1987), pp.25-59.

[4] F. Belina and D. Hogrefe, "The CCITT Specification and Description Language SDL", Computer Networks and ISDN Systems, Vol. 16, pp.311-341, 1989.

[5] Ed Brinksma, Giuseppe Scollo, and Chris Steenbergen, "LOTOS Specification, Their Implementations and Their Tests", IFIP Protocol Specification, Testing, and Verification VI, Ed. by B. Sarikya, and G.v. Bochmann, Elsevier Science Publishers B.V.(North-Holland), 1987, pp.349-360.

[6] Ed Brinksma, "A Theory for the Derivation of Tests", IFIP Protocol Specification, Testing, and Verification VIII, Ed. by S. Aggarwal and K. Sabnani, Elsevier Science Publishers B.V. (North-Holland), 1988, pp.63-74.

[7] Ed Brinksma, Rudie Alderden, Rom Langerak, Jeroen van de Lagemaat and Jan Tretmans, "A Formal Approach to Conformance Testing", the proceedings of 2nd International Workshop on Protocol Test Systems, (Berlin(West), Germany, Oct. 3-6, 1989), Ed. by J. de Meer, 1990, Elsevier Science Publishers B.V.(North-Holland), pp.349-363.

[8] S. Budkowski and P. Dembinski, "An Introduction to Estelle: A Specification Language for Distributed Systems", Computer Networks and ISDN Systems, Vol. 14, No.1, 1987, pp.3-23.

[9] E. Cerny, "Verification of I/O Trace Set Inclusion for a Class of Nondeterministic Finite State Machines", ICCD'92 Conference, Cambridge, Mass., October 1992.

[10] T.S.Chow, "Testing Software Design Modeled by Finite-State Machines, IEEE Transactions on Software Engineering, Vol. SE-4, No.3, 1978, pp.178-187.

[11] K.Drira, P.Azema, B.Soulas and A.M.Chemali, "Testability of a communicating system through an enviroment", TAPSOFT'93: Theory and Practice of Software Development (the Proceedings of the 4th International Joint Conference CAAP/FASE, 1993), Eds by M.-C.Gaudel, and J.-P.Jounnaud, Springer-Verlag, Berlin, pp.529-543.

[12] N. V. Evtushenko and A. F. Petrenko, "Fault-Detection Capability of Multiple Experiments", Automatic Control and Computer Science, Allerton Press, Inc., New York, Vol.23, No.3, 1989, pp.7-11.

[13] N. V. Evtushenko and A. F. Petrenko, "Method of Constructing a Test Experiment for An Arbitrary Deterministic Automaton", Automatic Control and Computer Science, Allerton Press, Inc., New York, Vol.24, No.5, 1990, pp.65-68.

[14] S.Fujiwara, G. v. Bochmann, F.Khendek, M.Amalou and A.Ghedamsi, "Test Selection Based on Finite State Models", IEEE Transactions on Software Engineering, Vol SE-17, No.6, June, 1991, pp.591-603.

[15] Susumu Fujiwara and Gregor v. Bochmann, "Testing Nondeterministic Finite State Machine with Fault Coverage", IFIP Transactions, Protocol Testing Systems IV (the Proceedings of IFIP TC6 Fourth International Workshop on Protocol Test Systems,1991), Ed. by Jan Kroon, Rudolf J. Heijink and Ed Brinksma, 1992, North-Holland, pp.267-280.

[16] S. Fujiwara and G. v. Bochmann, "Testing Nondeterministic Finite State Machine", Publication #758 of D.I.R.O, University of Montreal, January 1991.
[17] A. Gill, Introduction to the Theory of Finite-State Machines, New York: McGraw-Hill, 1962, 270p.
[18] G. Gonenc, "A Method for Design of Fault Detection Experiments", IEEE Transactions on Computer, Vol C-19, June, 1970, pp.551-558.
[19] C. A. R. Hoare, Communicating Sequential Processes, Prentice Hall, 1985.
[20] John E.Hopcroft, Jeffery D.Ullman, Introduction to Automata Theory, Languages, and Computation, 1979, Addison-Wesley Publishing Company, Inc., 418p.
[21] ISO, Estelle - A Formal Description Technique Based on an Extended Finite State Transition Model, IS 9074.
[22] ISO, Lotos - A Formal Description Technique Based on the Temporal Ordering of Observational Behavior, IS-8807, 1989.
[23] Hans Kloosterman, "Test Derivation from Nondeterministic Finite State Machines", IFIP Transactions, Protocol Test Systems,V (Proceedings of 5th International Workshop on Protocol Testing Systems, Montreal, Canada, 1992), Ed. by G.v. Bochmann, R. Dssouli and A. Das, 1993, North-Holland, pp.297-308.
[24] J. Kroon, Inres State Tables, Private Communication, 1992.
[25] D.Y. Lee and J.Y. Lee, "A Well-Defined Estelle Specification for the Automatic Test Generation", IEEE Transactions on Computers, Vol.40, No.4, April, 1991, pp.526-542.
[26] Gang Luo, Junliang Chen, "Generating Test Sequences for Communication Protocol Modeled by CNFSM", Information Technology: Advancement, Productivity and International Cooperation (Proc. of the 3rd Pan Pacific Computer Conference), Vol.I , Ed. by Chen Liwei et al, 1989, International Academic Publishers, pp.688-694.
[27] Gang Luo, Gregor von Bochmann, Anindya Das, and Cheng Wu, "Failure-Equivalent Transformation of Transition System to Avoid Internal Actions", Information Processing Letters, Vol. 44, No.6, December, 1992, Elsevier Science Publishers B.V.(North-Holland), pp.333-343.
[28] Gang Luo, Alexandre Petrenko, and Gregor von Bochmann, "Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines", Publication #864 of D.I.R.O, University of Montreal.
[29] Gang Luo, Gregor v. Bochmann, and Alexandre Petrenko, "Test Selection Based on Communicating Nondeterministic Finite State Machines Using A Generalized Wp-Method", IEEE Transactions on Software Engineering, Vol. SE-20, No.2, 1994, pp.149-162.
[30] Gang Luo, Anindya Das and Gregor v. Bochmann, "Software Testing Based on SDL with SAVE", IEEE Transactions on Software Engineering, Vol. SE-20, No.1, 1994, pp.72-87.
[31] Robin Milner, A Calculus of Communicating Systems, Lecture Notes in Computer Science Vol.92, 1980, Sringer-Verlag, 171p.
[32] S.Naito and M.Tsunoyama, "Fault Detection for Sequential Machines by Transition Tours", in Proc. of Fault Tolerant Comput. Syst., 1981, pp.238-243.
[33] Alexandre Petrenko, "Checking Experiments with Protocol Machines", IFIP Transactions, Protocol Testing Systems IV (the Proceedings of IFIP TC6 Fourth International Workshop on Protocol Test Systems, 1991), Ed. by Jan Kroon, Rudolf J. Heijink and Ed Brinksma, 1992, North-Holland, pp.83-94.
[34] Alexandre Petrenko and Nina Yevtushenko, "Test Suite Generation for a FSM with a Given Type of Implementation Errors", IFIP Transactions, (Proceedings of IFIP 12th International Symposium on Protocol Specification, Testing, and Verification, U.S.A., 1992), Ed. by R. J. Linn, Jr. and M. U. Uyar, pp.229-243.
[35] Alexandre Petrenko, Gregor von Bochmann and Rachida Dssouli, "Conformance Relations and Test Derivation", (invited paper), the Proceedings of IWPTS'93, 1993, Ed. by Omar Rafiq, pp.161-182.
[36] D. H. Pitt and D. Freestone, "The Derivation of Conformance Tests from Lotos Specifications", IEEE Transactions on Software Engineering, Vol.16, No.12, Dec. 1990, pp.1337-1343.

[37] D. Rayner, "OSI Conformance Testing", Comput. Networks & ISDN Syst., Vol.14, 1987, pp.79-89.
[38] Anne Bourguet-Rouger and Pierre Combes, "Exhaustive Validation and Test Generation in Elivis", SDL Forum'89: The Language at Work, North-Holland, 1989.
[39] K.Sabnani & A.T.Dahbura, "A Protocol Test Generation Procedure", Computer Networks and ISDN, Vol.15, No.4, 1988, North-Holland, pp.285-297.
[40] Behcet Sarikaya and Gregor v. Bochmann, "Synchronization and Specification Issues in Protocol Testing", IEEE Transactions on Communications, Vol.COM-32, No.4, April 1984, pp.389-395.
[41] B. Sarikaya, G.v. Bochmann, and E. Cerny, "A Test Design Methodology for Protocol Testing", IEEE Transactions on Software Engineering, Vol.13, No.9, Sept.. 1987, pp.989-999.
[42] D. P. Sidhu and T. K. Leung, "Formal Methods for Protocol Testing: A Detailed Study", IEEE Transactions on Software Engineering, Vol SE-15, No.4, April, 1989, pp.413-426.
[43] P. H. Starke, Abstract Automata, North-Holland/American Elsevier, 1972, 419p.
[44] Jan Tretmans, "Test Case Derivation from LOTOS Specifications", Formal Description Techniques, II, (Proceedings of the IFIP Second International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, 1989), Ed. by Son T. Vuong, 1990, North-Holland, pp.345-359.
[45] Piyu Tripathy and Behcet Sarikaya, "Test Generation from LOTOS Specification", IEEE Transactions on Computer, Vol C-40, No.4, 1991, pp.543-552.
[46] M. P. Vasilevskii, "Failure Diagnosis of Automata", Cybernetics, Plenum Publishing Corporation, New York, No.4, 1973, pp.653--665.
[47] S. T. Vuong, W.W.L. Chan, and M.R. Ito, "The UIOv-method for Protocol Test Sequence Generation", Proceedings of IFIP TC6 Second International Workshop on Protocol Testing Systems, Ed. by Jan de Meer, Lothar Machert and Wolfgang Effelsberg, 1989, North-Holland, pp.161-175.
[48] S. T. Vuong, A.A.F.Loureiro and S.T.Chanson, "A Framework for the Design for Testability of Communication Protocols", (invited paper), the Proceedings of IFIP TC6 Sixth International Workshop on Protocol Test Systems, 1993, Ed. by Omar Rafiq, pp.91-111.
[49] M. F. Witteman, R. C. van Wuijtswinkel and S.Ruud Berkhout, "Nondeterministic and Default Behaviour", IFIP Transactions, Protocol Test Systems, V (Proceedings of 5th International Workshop on Protocol Testing Systems, Montreal, Canada, 1992), Ed. by G.v. Bochmann, R. Dssouli and A. Das, 1993, North-Holland, pp.275-288.