13

Testing Strategies for Communicating FSMs

A. Petrenko, N. Yevtushenko✪, R. Dssouli

Université de Montréal, Canada
✪ Tomsk State University, Russia

The paper addresses grey-box testing of a system of communicating finite state machines under the assumptions that the system's structure is preserved in all potential implementations, and that at most one component can be faulty. Several possible testing strategies which rely on these assumptions in various manners are presented and compared. The ideas behind these strategies are explained on a simple serial composition of two communicating FSMs. The existing FSM-based test derivation methods are assessed for their applicability to solve the problems arising from grey-box testing.

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

1. INTRODUCTION

There have been many research efforts on conformance test derivation for protocols based on their formal models, mainly the LTS and FSM models, and the black-box representation of an implementation under test (IUT). According to the black-box strategy, a test suite is generated from, for example, a single FSM representing the reference behavior, and it is assumed that all possible faults are modeled by a finite set of mutant (faulty) FSMs [Petr93a]. This test suite is to be considered as complete, i.e. having complete fault coverage, if it can detect nonequivalence of any FSM from the predefined set of machines (the fault domain) to the reference (deterministic) FSM. It is natural to believe that as the power of this set increases, the length of the complete test suite increases as well. Thus, its length essentially depends on the accuracy of our choice of the set of mutants. In context of black-box testing, this set is usually defined by an upper bound on the number of states in all potential implementations, since no assumptions are made about the internal structure of an IUT [Gill62].

In practice, however, an IUT is often embedded within a complex system under test (see, e.g. the embedded test method of ISO [IS9646]), or it has some components that have been thoroughly tested in isolation. A similar situation arises when testing has to be performed based on the assumption that at most one component can be faulty in the known structure of the given compound implementation. A test engineer may impose this assumption because it is, in fact, plausible under the given circumstances, i.e. the probability of several components being faulty is realistically assumed to be much less than that of a single one. On the other hand, the engineer may wish to simply avoid test explosion at the price, perhaps, of a worse fault coverage. In the last case, he has in his support a widely acknowledged observation that tests for single faults usually capture multiple faults as well.

In all the above presented situations, a black-box representation of an implementation becomes inadequate for the amount of available information. Actually, this information could be used to refine the fault domain and reduce the number of tests. The related testing strategy is known as grey-box testing [Morr93] or structured testing [Lee93]. In particular, grey-box

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

testing is interpreted as a situation where the structure of the tested composed system is known, but the implementation details within each component remain hidden. As an example, testing concurrent systems for interoperability is essentially grey-box testing [Cast93]. Grey-box testing of FSMs was first considered in the context of functional testing of sequential circuits [Das75].

In this paper, we consider grey-box testing of a system of communicating finite state machines, assuming that its structure is preserved in all potential implementations and that at most one component can be faulty. We present and compare several possible testing strategies which rely on these assumptions in various manners. The ideas behind these strategies are explained on a simple serial composition of two communicating FSMs. The existing FSM-based test derivation methods are assessed for their applicability to solve the problems arising from grey-box testing.

The rest of this paper is structured as follows. Section 2 contains several notions related to the FSM model and introduces the problem of the grey-box testing of communicating FSMs. In Section 3, we analyze the testing strategy based on the composed machine construction as well as an ad hoc strategy based on the choice of a subset of transitions to be checked. Section 4 presents a strategy that relies on use of a so-called fault function which allows one to refine the fault domain needed for deriving a complete test suite. Section 5 presents a strategy that is based on the idea of transforming the given modular specification into a so-called testable representation of behavior of the component under test. Because of space constraints, the proofs are omitted, for more details, see [Petr94]. In Section 6, we compare these strategies based on the length of complete test suites obtained for a simple example which is used throughout the paper. Open issues are discussed in Section 7.

2. BASIC NOTIONS AND DEFINITIONS

2.1. NFSMs

A nondeterministic finite state machine (NFSM), often simply called a *machine* throughout this paper, is an initialized nondeterministic Mealy machine which can be formally defined as follows [Petr93a]. A *nondeterministic finite state machine* A is a 6-tuple (S, X, Y, h, s_0, D_A), where S is a set of n states with s_0 as the initial state; X - a finite set of input symbols; Y - a finite set of output symbols; D_A - a specification domain which is a subset of $S \times X$; h - a behavior function $h: D_A \rightarrow \mathbb{P}(S \times Y)$, and $\mathbb{P}(S \times Y)$ is the powerset of $S \times Y$.

Let $\alpha = x_1x_2...x_k \in X^*$, α is called an *acceptable input sequence for state* $s_i \in S$, if there exist k states $s_{i1}, s_{i2}, ..., s_{ik} \in S$ and an output sequence $\gamma = y_1y_2...y_k \in Y^*$ such that there is a sequence of transitions $s_i\text{-}x_1/y_1\text{->}s_{i1}\text{-}x_2/y_2\text{->}s_{i2}$ -> ... -> $s_{ik\text{-}1}\text{-}x_k/y_k\text{->}s_{ik}$ in the machine. We use X_i^* to denote the set of all the acceptable input sequences for the state s_i and X_A^* for the state s_0, i.e. for A.

An NFSM A is said to be *completely specified*, if $D_A = S \times X$. We will omit the specification domain D_A in the case of completely specified machines. If D_A is a proper subset of $S \times X$ then A is considered *partially specified*. An NFSM can be referred to as a complete or a partial machine.

We extend the behavior function to the set X_A^* of all acceptable input words (sequences) containing the empty word e, i.e., $h: S \times X_A^* \rightarrow \mathbb{P}(S \times Y^*)$. The function h^1 is the *next state* function, while h^2 is the *output* function of NFSM A [Petr93a].

The machine A becomes *deterministic* when $|h(s,x)| = 1$ for all $(s,x) \in D_A$. In a deterministic FSM, instead of the behavior function, we use two functions: the next state function δ, and the output function λ.

The *equivalence* relation between two states s_i and s_j in the NFSM A holds if

(i) $X_i^* = X_j^*$ and (ii) $\forall \alpha \in X_i^*$ ($h^2(s_i,\alpha) = h^2(s_j,\alpha)$), otherwise, they are nonequivalent.

Two NFSMs A and B are said to be *equivalent* if their initial states are equivalent.

We will be interested in machines which are initially connected. Given a NFSM $A = (S, X, Y, h, s_0, D_A)$, A is said to be *initially connected* if $\forall s \in S\ \exists \alpha \in X_A^*$ $(s \in h^1(s_0,\alpha))$. Every NFSM is equivalent to an initially connected one.

The complete NFSM $B = (T, X, Y, H, t_0)$ is said to be *quasi-equivalent to A* if for all acceptable input sequences $\alpha \in X_A^*$ ($H^2(t_0,\alpha) = h^2(s_0,\alpha)$).

Given the NFSM $A = (S, X, Y, h, s_0, D_A)$, and complete NFSM B, B is said to be a *reduction* of A, written $B \leq A$, if

$$\forall \alpha \in X_A^*\ (H^2(t_0,\alpha) \subseteq h^2(s_0,\alpha)).$$

If $B \leq A$ and B is deterministic then it is referred to as a *D-reduction* of A.

An NFSM $B = (S', X, Y, h', s_0)$ is said to be a *submachine* of the NFSM $A = (S, X, Y, h, s_0)$ if $S' \subseteq S$ and $h'(s,x) \subseteq h(s,x)$ for all $(s,x) \in S' \times X$. Obviously, all submachines of A are its reductions. If a submachine of A is deterministic then it is said to be a *D-submachine* of A.

We will use the reduction and quasi-equivalence relations between NFSMs as the conformance relations between implementations and their specifications which are essential to deriving test suites [Petr93a]. We assume that all potential faults are represented by a finite set of "mutant" NFSMs of the reference machine A [Petr93a], i.e., the set $\mathfrak{I}$ of completely specified NFSMs with the same input alphabet X is the fault model. If this set is a universal set of all machines with at most m states then we denote it $\mathfrak{I}_m$.

Let E be a finite set of finite input sequences from X_A^*. A test suite E is said to be *complete* for the specification NFSM A w.r.t. the reduction relation in the class $\mathfrak{I}$ if for any NFSM $B = (T, X, Y, H, t_0)$ in this set $\mathfrak{I}$ which is not a reduction of A

$$\exists\, \alpha \in E\ \exists \gamma \in H^2(t_0,\alpha)\ \ (\gamma \notin h^2(s_0,\alpha)).$$

In words, for every possible mutant NFSM that is not a reduction of the NFSM A, the test suite E should have at least one input sequence which eliminates it. Note that in the rest of the paper, we assume a reliable reset feature in potential implementations.

A test suite E is said to be *complete* for the NFSM A w.r.t. the quasi-equivalence relation in the class $\mathfrak{I}$ if, for any $B = (T, X, Y, H, t_0)$ in this set $\mathfrak{I}$ which is not quasi-equivalent to A

$$\exists\, \alpha \in E\ (H^2(t_0,\alpha) \neq h^2(s_0,\alpha)).$$

A complete test suite guarantees complete coverage of all faults from the predefined domain $\mathfrak{I}$.

2.2. Serial composition of NFSMs and its testing

Consider testing a component of a system C (Figure 1) which is the serial connection of two complete NFSMs, $H = (S, X, Z, h_1, s_0)$ and $T = (P, Z, Y, h_2, p_0)$.

Suppose a next input symbol is only submitted to the system after it has produced an output symbol in response to the previous input symbol. This means the behavior of the system C can be specified by an NFSM $A_C = (S \times P, X, Y, h, s_0p_0)$ further referred to as a *composed* machine, where the behavior function h is defined as follows. For any $(sp,x) \in S \times P \times X$

$$h(sp,x) = \{\ (s'p',y) \mid s' \in h_1^1(s,x),\ (p',y) \in h_2(p,z),\ (s',z) \in h_1(s,x)\}.$$

The defined composed machine could have equivalent states as well as states that are not reachable from the initial state. If we are only interested in analyzing the behavior the whole system then we can always merge all equivalent states and delete unreachable states.

Two systems are said to be *equivalent* if their corresponding NFSMs are equivalent. The system C is a *reduction* of the system D if the NFSM A_C is a reduction of the NFSM A_D.

Now, we define a fault model and complete test suite for the composed NFSM A_C of the given system C under the assumption that only one component, either H or T, can be faulty.

Let $\mathfrak{I}_m(C,H)$ be a set of all composed NFSMs which correspond to the systems obtained from the given system C by replacing the component H with a mutant of $\mathfrak{I}_m$, which is the set of all mutants of H, $m \geq |S|$. A *complete* test suite *for a component* H of C in the class $\mathfrak{I}_m$ w.r.t. the reduction (equivalence) relation is now defined as a complete test suite for the NFSM A_C in the class $\mathfrak{I}_m(C,H)$ w.r.t. the reduction (equivalence) relation. A fault model $\mathfrak{I}_m(C,T)$ and complete test suite for a component T are similarly defined.

In the remaining part of this paper, we consider compositions of only deterministic machines to simplify our discussion on grey-box based testing strategies.

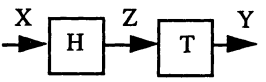

Figure 1: The system C.

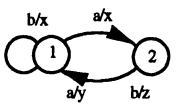

Figure 2: The FSM H.

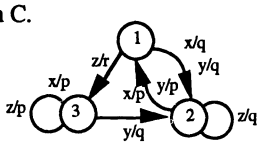

Figure 3: The FSM T.

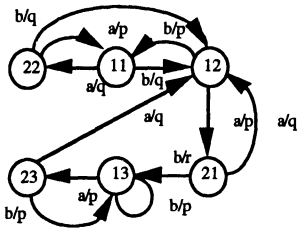

Figure 4: The composed FSM.

Example. Consider the system shown in Figure 1. Assume the behavior of the head component is specified by the FSM H given in Figure 2, and that of the tail component by the FSM T in Figure 3. Following the rules for constructing a composed machine given above, we obtain the machine in Figure 4. The initial state of this machine is 11, since the initial states of H and T are assumed to be 1 and 1. This machine corresponds to an "ideal" case when a composed machine has no equivalent or unreachable states.

■

We will use this simple example to compare the length of test suites derived, when different methodologies are used for testing this system under the assumption that only one component can be faulty. Such a testing is referred to as *grey-box* testing of the given system, in contrast to black-box testing, which requires no knowledge of the internal IUT structure, or assumptions of fault-free components.

This system can be easily tested on a component by component basis with the black-box strategy applied to each isolated component, provided that a point of control and observation (PCO) is introduced into the system. Since this is not always feasible, we further consider the testing strategies which do not rely on the existence of PCOs. At the same time, we may interpret the composition as a part of some modular system with the point of control at the input of H and the point of observation at the output of T.

Concluding this section, we derive for our example, two complete test suites for the components tested in isolation, in order to have a common basis for comparing length of various test suites obtained by following the different strategies. We assume that no fault can increase the number of states in implementations of a reference machine and apply a method such as the Wp-method [Fuji91]. For the isolated FSM H, the complete test suite in the class of $\mathfrak{I}_2$ is { *raaa, raba, rba* }.

It has length of 11 (resets *"r"* are also counted). The complete test suite for the isolated FSM T in the class $\mathfrak{I}_3$ has length 27:

{ *rxz, ryxz, ryyz, ryzz, rzxz, rzyz, rzxz, rzzz* }.

Comparing the length of the above given test suite with that of a test suite needed to test the corresponding component embedded into the given system, we could see how its testability deteriorates.

3. TESTING THROUGH THE COMPOSED MACHINE CONSTRUCTION

3.1. Black-box testing of the composed machine

In order to finally exploit a traditional FSM-based test derivation method [Petr93a], [Sidh89], we can replace the grey-box view of the given system by the composed machine. This machine now serves as a reference machine for deriving test suites, provided an upper bound on the number of states in the implementation under test is known, i.e. the fault model is defined [Luo94a]. Unfortunately, even if it is assumed that the faults in any implementation of any component do not increase its number of states, a corresponding composed machine may still have a larger number of states than the reference composed machine. There may exist unreachable or equivalent states in the reference machine, which may become reachable or distinct in the mutant composed machine. In the worst case, all $|S \times P|$ states are reachable and distinct, so a test suite has to be constructed in the class $\mathcal{S}_m$, where $m = |S \times P|$. At the same time, consider the example where faults in the head component can increase the number of states from $|S|$ to $|S'|$. It does not necessarily imply that $|S' \times P|$ should be taken as a proper upper bound. A more rigorous analysis is needed to identify the exact maximal number of states in all possible mutants, assuming that faults are only located in a single component. The actual goal is the reduction of a final test suite, since it is well-known that the size of a test suite required to test a machine with m states with respect to a reference machine with n states ($n \leq m$) grows exponentially as $(m-n)$ increases [Vasi73], [Chow78], [Fuji91]. A more exact fault model would usually yield a shorter complete test suite.

Example. To illustrate the results of treating the above constructed composed FSM as a black-box, we derive a complete test suite for the composed machine (Figure 4) in the class of all FSMs with at most 6 states, assuming that no fault can increase the number of states in any component. The Wp-method [Fuji91] gives the following result. The state cover set is $V = \{e, a, b, ba, bab, baba\}$. The characterization set is $W = \{a, bb\}$; based on this set, we can construct the following state identifiers: $W_1 = \{a,b\}$, $W_2 = \{bb\}$, $W_3 = \{a,bb\}$, $W_4 = \{b\}$, $W_5 = \{a,b\}$, $W_6 = \{a,b\}$. The resulting test suite contains 11 test sequences of total length 63:

{ *raaa, raab, rabbb, rbba, rbbb, rbaabb, rbabaabb, rbababa, rbababbb, rbabba, rbabbbb* }.

To test both components in isolation we need only 38 test events as shown in Section 2.2.

■

A test suite derived in this way from a reference composed machine usually contains a large amount of redundancy, since it neglects to exploit the assumption that all components but one, and the interconnection between them, are correctly implemented. More sophisticated strategies are required to reduce this kind of redundancy. In the remaining part of the paper, we present some of them and compare the length of the resulting test suites for the same example presented in Figures 1 - 4.

Summarizing the above discussion we conclude that a straightforward application of the black-box testing methodology to the grey-box testing:

- requires a precise analysis of the effect of faults on the maximal number of states in potential implementations;
- relies upon the existing test derivation methods once that upper bound is established;
- faces the problem of test explosion;
- results in a test suite with a certain amount of redundancy which should and could be avoided.

In the following sections we elaborate certain strategies which take more advantage of the knowledge of the internal structure and the assumption of a single faulty component.

3.2. Checking transitions in the composed machine

Since the states of a composed machine are structured and the transitions between states are composed of transitions of component machines, it is natural to try (see, for example,

[Dam91]) to construct a complete test suite for the composed machine A_C in such a way that only an appropriate subset of the transitions in A_C are checked. An implied approach, however, heavily depends on what is meant by saying "a transition is, in fact, checked". The test derivation methods such as the W- [Vasi73], [Chow78], Wp- [Fuji91], UIOv- [Vuon89] methods foresee the two distinct phases of testing a machine under test, namely: the state verification phase and transition checking phase; others, like [Petr91], [Petr92], do so implicitly. Omitting the former makes the latter indeterminate; one can no longer state that a transition has then been checked (see the related discussion in [Vuon89] on the UIO-based methods). Consequently, we further base our discussion on checking a restricted number of transitions on existence of the state verification phase.

Clearly the choice of an appropriate subset of transitions in a composed machine depends to a large extent on the manner in which the output signal of component H is propagated to the external output. As can be seen from Figures 2 and 3, certain errors in H which are distinguishable at the output of H are masked by T. For example, component T in state 1 processes its inputs x and y in the same manner. Thus, if the composed machine is in state 11, the erroneous output y of component H in response to the input a will not be detected at the output of the system; this particular error can be detected if the composed machine is in state 13. However, an erroneous output z of component H in response to the input a will not be detected at the external output if the composed machine is in state 13, but it will be detected if the system is in state 11, reversing the roles played by these two states.

The above example clearly shows that in certain cases, it is not enough to check a transition in component H once only. In fact, component T in Figure 3 has been so constructed to demonstrate this fact. Note that in this example, there are two transitions in the head component, from state 2, that can be tested only once in the composed machine, but not in an arbitrary its state. For example, the transition from state 2 under input a cannot be tested only once if the composed machine is in state 22. However, in a complex system, such an analysis becomes rather cumbersome.

Example. Based on the above discussion, we construct a complete test suite for this particular example in an *ad hoc* manner. The idea behind this construction is to use the characterization set $W = \{a, bb\}$ for state identification, and state identifiers for checking tail states of transitions chosen in an appropriate way. In particular, assuming that only the head component can be faulty, i.e. the tail component is fault-free, it is sufficient to check the following transitions: 11-*a/q*->22, 11-*b/q*->12, 13-*a/p*->23, 13-*b/p*->33, 23-*a/q*->12, 22-*b/q*->12. The test suite has 8 test sequences of total length 44:

{ *raa, rabbb, rbaa, rbabaabb, rbababb, rbabba, rbabbbb, rbbb* }.

In a similar way, we derive a test suite assuming that only the tail component can be faulty. We obtain 9 test sequences of total length 52:

{ *raaa, raab, rabbb, rbaabb, rbabaabb, rbababa, rbababbb, rbabbb, rbbb* }.

If we now merge these test suites, we obtain the test suite which can reveal faults located in either the head or the tail component:

{ *raaa, raab, rabbb, rbaabb, rbabaabb, rbababa, rbababbb, rbabba, rbabbbb, rbbb* }.
The test suite has length of 59, i.e. slightly shorter that the one constructed in Section 3.1. ■

We call such an approach the *ad hoc* strategy. This strategy can be characterized as follows:
- it may give better results than the black-box strategy, since a complete test suite can always be obtained by deleting, in an ad hoc manner, some (sub) sequences from the test suite constructed by the last strategy;
- a very rigorous analysis is needed to guarantee the complete fault coverage;
- it is not easy to automate.

Next we consider more formalized grey-box based testing strategies.

4. THE USE OF A FAULT FUNCTION

As mentioned above, test derivation from the reference composed machine in the class $\mathfrak{I}_m$ results in redundant tests. The main reason for this is that this fault domain includes mutants that should be excluded under a single faulty component assumption. Consider, for example, the transition 11-*b/q*->12 in the composed machine A_C (Figure 4). If all the potential faults are in the tail component, then it does not make sense to distinguish the reference machine from mutants having transitions from 11 to 21, 22, or to 23 under *b/q*. In fact, since the head component does not have any errors, the composed machine cannot enter those states due to faults in the tail component. The conclusion is that the fault domain has to be refined to get a shorter, but still complete test suite.

The fault domain can be specified exactly by explicitly enumerating all the mutant machines of the constructed composed machine. In this case, the fault model would actually be a list of FSMs as, for example, in [Brzo91]. However, the number of mutants tends to explode, and therefore there is a need for a more compact representation of mutants with restricted faults which may occur in a composed machine. An appropriate method should tune a test suite to cover these mutants exclusively. So far, there is only one such method, namely, the FF-method [Petr92] that is based on the notion of a fault function. The method partially meets these requirements. In the next section, we recall the notion of a fault function from [Petr92].

4.1. The fault function

Let $A = (S, X, Y, \delta, \lambda, s_0)$ be a complete reference FSM. To model potential deviations of the IUT behavior from the reference FSM or, in other words, the mutants of A, a function F is introduced describing their range. The function $F: S' \times X \to \mathbb{P}(S' \times Y)$, where $S' \supseteq S$, and $\mathbb{P}(S' \times Y)$ is the set of all subsets from $S' \times Y$, is called a fault function for the given reference machine A, if

$$\forall (s,x) \in S \times X \ ([\delta(s,x), \lambda(s,x)] \in F(s,x)).$$

Thus, to represent the superfluous states implemented by mistake or due to other reasons, the set $S' \setminus S$ is used. The set $F(s_0) \subseteq S'$ represents all possible mutant initial states.

We associate with the fault function F a set $F(A)$ of deterministic FSMs which are mutants of the given reference machine A, i.e.

$$F(A) = \{B=(S', X, Y, \Delta, \Lambda, s_{0B}) \mid s_{0B} \in F(s_0) \ \& \ \forall (s,x) \in (S \times X) \ ([\Delta(s,x), \Lambda(s,x)] \in F(s,x)) \}.$$

The set $F(A)$ defines a fault domain for the reference machine A. The quintuple $A_F = (S', X, Y, F, F(s_0))$ is, in fact, a (weakly initialized [Star72]) NFSM with the set of initial states $F(s_0)$, and the defined function F specifies exactly its reaction to input words. It is conceded that $F(s,\alpha)$ comprises, in the general case, states and output words that are not the reactions of any deterministic machines from $F(A)$. The following relations between the reactions of the reference FSM A, machine A_F and a joint reaction of the mutants from $F(A)$ hold:

$$[\delta(s,\alpha), \lambda(s,\alpha)] \in \{ [\Delta(s,\alpha), \Lambda(s,\alpha)] \mid (S', X, Y, \Delta, \Lambda, s_{0B}) \in F(A) \} \subseteq F(s,\alpha).$$

The use of the fault function allows us to refine the fault domain for the composed machine. In particular, we next show how a fault function for the composed machine can be formally defined based on a single faulty component assumption.

4.2. Constructing the fault function for the faulty head machine

Now we describe the construction of the fault function for the given composed machine under the assumption that only the head component can be faulty. Let $H = (S, X, Z, \delta_1, \lambda_1, s_0)$ and $T = (P, Z, Y, \delta_2, \lambda_2, p_0)$ be the component machines of C. The idea of the construction is as follows. Consider a transition in the composed machine from state sp under

the input x. As a result of the head component's malfunctioning, the composed machine can enter some state $s'p'$. The first component state s' is any state of the head machine. The second state p' is a successor of the state p in the tail machine for any its input, producing the output corresponding to the transition from p into the successor p'. Formally, the fault function is specified for any $sp \in S \times P$ and $x \in X$:

$$F(sp, x) = \{ (s'p', y) \mid s' \in S \ \& \ (p', y) = (\delta_2(p,z), \lambda_2(p,z)) \text{ for all } z \in Z \}, \ F(s_0p_0) = S \times \{p_0\}.$$

Example. Table 1 gives the fault function for our example when the tail component is assumed to be fault-free. $F(11) = \{11,21\}$. Bold symbols in the table correspond to the reference composed machine (Figure 4).

Table 1: The fault function for the faulty head component.

input	11	12	13	21	22	23
a	**22,12/q** 13,23/r	**21,11/p** 12,22/q	**23,13/p** 12,22/q	**12,22/q** 13,23/r	**11,21/p** 12,22/q	**12,22/q** 13,23/p
b	**12,22/q** 13,23/r	**11,21/p** 12,22/q	**13,23/p** 12,22/q	**13,23/r** 12,22/q	**12,22/q** 11,21/p	**13,23/p** 12,22/q

This table is interpreted also as a state table of the NFSM C_F which characterizes all the mutants, i.e. the fault domain, of the composed machine, under the assumption that the tail component is fault-free.

■

Proposition 4.1. Let C_F be an NFSM defined by the constructed fault function F. Any mutant of the composed machine with a faulty head machine is a D-submachine of C_F.

■

However, the converse is not true, since not every its D-submachine is a mutant of the composed machine with a faulty head machine. Consider Table 1 and a mutant composed machine with the transition from state 11 to state 12 under input a. Since the faulty head component remains in the same state 1 under input a, this machine cannot have a transition, for example, from the state 12 to state 21 under this input; however, such a machine is included in the fault domain $F(A_C)$.

As shown in [Petr92], the fault function provides a compressed representation of the set of mutants of the given reference machine. A test suite complete w.r.t. these restricted faults can be derived by applying the so-called FF-method, presented in the same paper.

Finally, we use this method to derive a complete test suite for the composed machine (Figure 4) and the fault function in Table 1. The test suite has 37 test events:

{ *raab, rabb, rbaab, rbabaab, rbababa, rbabba, rbbb* }.

4.3. Constructing the fault function for the faulty tail machine

In this section, we describe the construction of the fault function G for the given composed machine under the assumption that only the tail component can be faulty.

The idea behind this construction is similar to the one in the previous section. Consider a transition in the composed machine from state sp under the input x. As a result of the malfunctioning of the tail component, the composed machine can enter any state $s'p'$. The first component s' is the successor state of the head machine under this input. The second component p' is any state of the tail machine, producing any output. Formally, the fault function G is specified for any $sp \in S \times P$ and $x \in X$:

$$G(sp, x) = \{ (s'p', y) \mid s' = \delta_1(s,x), \ p' \in P, \ y \in Y \}, \ G(s_0p_0) = \{s_0\} \times P.$$

Example. Table 1 gives the fault function for the composition. $G(11) = \{11,12,13\}$. Bold symbols in the table corresponds to the reference composed machine (Figure 4).

Table 2: The fault function for the faulty tail component.

input	11	12	13	21	22	23
a	**22**,23,21 /**q**, p, r	**21**,22,23 /**p**, q, r	**23**,21,22 /**p**, q, r	**12**,11,13 /**q**, p, r	**11**,12,13 /**p**, q, r	**12**,11,13 /**q**, p, r
b	**12**,11,13 /**q**, p, r	**11**,12,13 /**p**, q, r	**13**,11,12 /**p**, q, r	**13**,11,12 /**r**, p, q	**12**,11,13 /**q**, p, r	**13**,11,12 /**p**, q, r

■

The corresponding NFSM C_G describes the fault domain in this case, as stated by the following proposition.

Proposition 4.2. Let C_G be an NFSM defined by the constructed fault function G. Any mutant of the composed machine with a faulty tail machine is a D-submachine of C_G.

■

Example. We derive a complete test suite for the composed machine (Figure 4) and the fault function in Table 2 using the FF-method. The resulting complete test suite has 42 symbols:

{ *raab, rabbb, rbaabb, rbabaabb, rbababbb, rbabbbb, rbbb* }.

Finally, we merge these test suites to obtain the test suite which can reveal faults located in either the head or the tail component:

{ *raab, rbababa, rbabba, rabbb, rbaabb, rbabaabb, rbababbb, rbabbbb, rbbb* }.

It has length of 55, i.e. it is shorter that the ones generated by the Wp-method with 63 symbols and by the ad hoc method with 59 symbols.

■

We call this presented approach to testing the composed system the *FF-based* strategy. Its main features are as follows:

- it provides a compact representation of mutants with faults which may occur in a limited number of components of the composed system;
- the resulting complete test suites are less redundant than in the case of black-box testing since the fault domain is more accurately determined;
- it is based on the formal method as opposed to the ad hoc strategy.

As shown above, this strategy yields such a test suite that covers not only all the potential faults but also the ones resulting in a nondeterministic behavior the system. Based on the assumption that any implementation of the given deterministic system also can be represented as a deterministic FSM, we conclude that the test suite might still be redundant. A strategy which takes care of this situation could give a better result. In the next section, we present such a strategy.

5. TESTABLE REPRESENTATION OF A COMPONENT UNDER TEST

When a component is connected as part of a compound system, the ability to control its inputs and observe its outputs is decreased. The state table of the isolated component does not characterize its behavior in the system. This problem was addressed in design of sequential circuits, and some procedures for modifying the state tables of components were proposed [Kim72]. The modified FSM is a specification of the component's behavior such that is controllable from the external inputs and observable at the external outputs of the whole system. In the context of design of sequential circuits, the modified FSM is used for achieving a further optimization of the given system of interacting FSMs [Rho91], [Deva89]. Here, we further develop this approach for constructing such *"testable"* representations of the components that can be used for test derivation [Petr93b].

5.1. A component with nonobservable output

If the head component is faulty, then in response to a certain input sequence, it can produce an output sequence which does not coincide with the output sequence of H. However, the failure on the external output occurs if the tail component reacts differently to these input sequences; otherwise the fault may go undetected. As shown below, the FSM formalism will provide an elegant characterization of externally undetectable deviations in the behavior of an imbedded component.

Two input sequences of the FSM T α and γ are said to be *T-equivalent*, if T produces the same output sequence when these sequences are applied to its initial state. The relation of T-equivalence partitions the set of all input sequences of T into the equivalence classes. If an output reaction of the faulty head machine and the reaction of the reference head machine lead to the same externally observed output reaction in response to any input sequence, then there is no experiment (test) which could detect the faulty head component.

Two FSMs *H1* and *H2* are said to be *T-equivalent* if their output reactions to any input sequence are T-equivalent. The T-equivalence of different head components implies the equivalence of the composed machines, since their compositions with T have the same behavior.

Proposition 5.1. The two composed FSMs *C1* and *C2* with the same tail FSM T are equivalent iff their head FSMs are T-equivalent.

■

Now we introduce an inductive rule for constructing the set of input sequences of T which are T-equivalent to an output response γz of the FSM H when an input sequence αx is applied. Suppose such a set is already defined for γ, and let β be a sequence T-equivalent to γ. We must determine which symbol z' can extend the sequence β in order to form a T-equivalent sequence to γz. It is known that H enters state s under α, and T enters state p_1 under γ, i.e. the state of the composed machine is now $sp_1 = \delta(s_0p_0, \alpha)$. The output of the composed machine is $\lambda(sp_1,x)$. Assume T enters state p_2 under the sequence β. Then $\beta z'$ is T-equivalent to γz if and only if $\lambda(sp_1,x) = \lambda_2(p_2,z')$.

The above described rule for constructing the set of sequences leads us to the formal specification of a nondeterministic FSM $\hat{H}$ that contains all the sequences which are T-equivalent to the output response γ of the FSM H to any input sequence α.

Let $A_C = (S\times P, X, Y, \delta, \lambda, s_0p_0)$ be the composed machine. Then $\hat{H} = (S\times P\times P, X, Z, h, s_0p_0p_0)$, where for any $x\in X$, $sp_1p_2\in S\times P\times P$, the behavior function is specified as follows: $h(sp_1p_2,x)$ is equal to $\{ (s'p'_1p'_2,z) \mid s'p'_1 = \delta(sp_1,x),\ z\in Z$ and $\lambda_2(p_2,z) = \lambda(sp_1,x) \}$ if there exists a z such that $\lambda_2(p_2,z) = \lambda(sp_1,x)$, otherwise it remains undefined.

Example. The NFSM $\hat{H}$ obtained from the above example is shown in Figure 5. A composed machine constructed from $\hat{H}$ and the machine T is isomorphic to the composed machine A_C (Figure 4). In other words, given the composed system, no experiment can distinguish which of the machines, the FSM H, or any reduction of $\hat{H}$ is used as the first component.

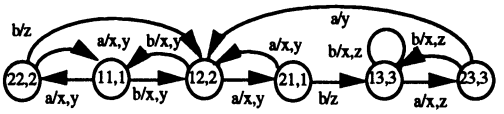

Figure 5: The NFSM $\hat{H}$

■

Proposition 5.2. Let $\widehat{H}$ be the NFSM constructed for the head FSM H of the composition C and let A_L be the composed FSM of a composition L with the same tail FSM T. A_L is equivalent to the composed FSM A_C iff the head FSM of L is a D-reduction of the NFSM $\widehat{H}$.

■

The set of D-reductions of the NFSM $\widehat{H}$ can have machines which are not equivalent to the head FSM H. These FSMs correspond to the implementations that would not conform to the isolated FSM H, but when considered in the composition C, become conforming to H. In other words, the NFSM $\widehat{H}$ is a complete characterization of all implementations with undetectable faults. Note that the problem of the characterization of conforming implementations w.r.t. the environment in terms of LTSs was recently discussed in [Drir93], and in terms of FSMs was independently considered in [Petr93b].

Proposition 5.3. Any complete test suite for the NFSM $\widehat{H}$ w.r.t. the reduction relation in the class $\mathfrak{I}_m$ is also a complete test suite for the component H of C w.r.t. the equivalence relation in the same class; and conversely, any complete test suite for the component H of C w.r.t. the equivalence relation in the class $\mathfrak{I}_m$ is also a complete test suite for the NFSM $\widehat{H}$ w.r.t. the reduction relation in the same class.

■

This proposition states that in order to test the deterministic head component in the serial composition, we have to derive a test suite from a nondeterministic FSM. The problem of test derivation in the grey-box context is thus transformed to a well-known problem of deriving tests in the black-box context, using a nondeterministic FSM and the reduction relation. So far, there are very few formal methods for solving the latter problem [Yevt91], [Petr93b].

Example. We now continue our example by deriving a complete test suite for the NFSM $\widehat{H}$ in Figure 5 with a version of the SC-method [Petr93b]. We know that all implementations of the specification NFSM $\widehat{H}$ are deterministic machines, so, assuming that faults in component H do not increase the number of states, we conclude that a complete test suite has to be derived for the class $\mathfrak{I}_2$. The resulting test suite has length of 22: { *raabaa, raaa, raabb, raba, rba* }.

■

Note that in spite of the fact that any input sequence is acceptable for the constructed NFSM $\widehat{H}$, this machine can be, in general, partially specified, which makes test derivation even more difficult for this class of nondeterministic machines.

This section demonstrates that nondeterministic machines provide a formal tool to describe potential deviations in the behavior of an imbedded deterministic component which do not alter the externally observed behavior.

5.2. A component with noncontrollable input

Two FSMs *T1* and *T2* are said to be *H-equivalent* if their output reactions to any output sequence of H coincide. The H-equivalence of different tail components implies the equivalence of the composed machines.

Proposition 5.4. The two composed FSMs *C1* and *C2* with the same head FSM H are equivalent iff their tail FSMs are H-equivalent.

■

The FSM T, once it is completely specified, can accept any input sequence from its input alphabet, however, the FSM H does not at all guarantee to deliver all these sequences. Therefore, we need to find such a "portion" of the behavior of T that can be enabled by H. The intuition behind is that only this part of T is of importance to the composed behavior. Even if the remaining part is faulty, there is no impact on the composed behavior. Since it cannot be externally tested anyway, it should be excluded from testing.

To proceed in this way, we first construct an acceptor to specify all the output sequences of the FSM H. Next, suppose we run this acceptor in parallel to the FSM T in such way that the acceptor signals to a current state of T whether its behavior is defined for a current input.

Formally, a (nondeterministic) acceptor is defined over the alphabet Z and the state set S with s_0 as the initial state, its next state function Φ' is specified as follows:

$$\Phi'(s,z) = \{\ s' \mid \exists\ x \in X\ (\ \delta_1(s,x) = s'\ \&\ \lambda_1(s,x) = z\ )\ \}$$

if the FSM H in state s can produce output z, otherwise $\Phi'(s,z)$ is left undefined. All its states are accepting states. If the acceptor is nondeterministic, then applying the standard technique [Hopc79] will yield a deterministic acceptor $A_Z = (B, Z, \Phi, b_0)$.

Now we are able to formally define a direct product $\hat{T} = (P \times B, Z, Y, \Delta, \Lambda, p_0 b_0, D)$ of the acceptor A_Z and the FSM T which corresponds to their parallel composition.

The specification domain D is the set $\{\ (pb, z) \mid \Phi(b,z)$ is defined$\}$. The functions of $\hat{T}$ are:

$$\Delta(pb,z) = (\delta_2(p,x),\ \Phi(b,z));\quad \Lambda(pb,z) = \lambda_2(p,x) \quad \text{for all } (pb, z) \in D.$$

This constructed FSM represents the testable behavior of the tail component, in the sense that all input sequences which determine this behavior can be delivered to the input of the component through the head machine. Moreover, $\hat{T}$ and T produce the same output responses to these sequences. The above procedure for obtaining the partial FSM (PFSM) $\hat{T}$ is similar to the one presented in [Kim72].

Example. The deterministic acceptor for the head component (Figure 2) is given in Figure 6. Figure 7 gives the PFSM $\hat{T}$ for our example.

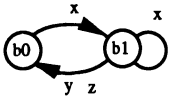

Figure 6: The acceptor A_Z

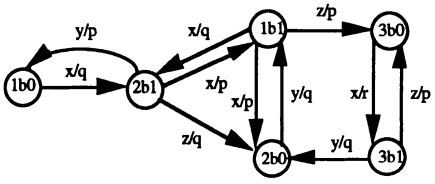

Figure 7: The PFSM $\hat{T}$

■

Proposition 5.5. Let $\hat{T}$ be the PFSM constructed for the tail machine T of the composition C and let A_L be the composed FSM of a composition L with the same head FSM H. A_L is equivalent to the composed FSM A_C iff the tail FSM of L is quasi-equivalent to the PFSM $\hat{T}$.

■

In general, the set of machines that are quasi-equivalent to the PFSM $\hat{T}$ can have machines which are not equivalent to the tail FSM T. These FSMs correspond to the implementations that would not conform to the isolated FSM T, but when considered in the composition C, they become conforming to T in C. In other words, the PFSM $\hat{T}$ is a complete characterization of all implementations with undetectable faults.

Any test suite for the PFSM $\hat{T}$ has the property that it is always possible to find a corresponding test suite for the whole system which produces the test suite for $\hat{T}$ on the internal input of the tail component. We say in this case, that there is a *translation* of one test suite into another.

Proposition 5.6. Any complete test suite for the PFSM $\hat{T}$ w.r.t. the quasi-equivalence relation in the class $\mathfrak{I}_m$ is translated into a test suite which is complete for the component T of C

w.r.t. the equivalence relation in the same class; conversely, any complete test suite for the component T of C w.r.t. the equivalence relation in the class $\mathfrak{I}_m$ is translated into a complete test suite for the PFSM $\widehat{T}$ w.r.t. the quasi-equivalence relation in the same class.

■

Since this proposition states that in order to test the tail component in the serial composition we must derive a test suite from a partial FSM, the problem of test derivation in the grey-box context is thus transformed to a well-known problem of deriving tests in the black-box context, from a partial (possibly unreduced) FSM w.r.t. the quasi-equivalence relation. However, almost all of the currently available methods are based on the so-called completeness assumption (see, for example, [Petr93a]), which is useless in this case. The unacceptable for $\widehat{T}$ input sequences cannot be delivered to the tail component through the head component and there is no way to test the behavior of the tail machine caused by these sequences. There are, however, test derivation methods [Yevt90], [Petr91], [Luo94] which do not rely on this assumption, and can be applied to partially specified and unreduced FSMs.

Example. We continue our example by deriving a complete test suite for the PFSM $\widehat{T}$ (Figure 7) in the class $\mathfrak{I}_m$ with the method [Yevt90], [Petr91]. The result is:

{ *rxxxx, rxxyxx, rxxzxx, rxxzxyx, rxxzxzx, rxyx, rxzx* }.

Then, we translate this test suite into a set of input sequences of the FSM H which cause them to occur on the input of the FSM T:

{ *rbbbb, rbaabb, rbabbbb, rbabaab, rbababb, raab, raba* }.

This test suite has 40 test events. Finally, we merge this test suite and the one derived in Section 5.1.1 and obtain the test suite which can reveal faults located in either the head or the tail component and has length of 51:

{ *raaa, raabaa, raabb, raba, rbaabb, rbabaab, rbababb, rbabbbb, rbbbb* }.

5.3. Summary of the presented strategy

As follows from the above discussion on transforming FSM-specifications, there is a new testing strategy based on the testable representations, which at the moment seems very promising and worth further research. In short, this strategy:

- gives a complete characterization of the implementations that do not conform to the isolated component, but when considered in the composition, they become conforming;
- provides the exact specification of the fault domain needed for test derivation;
- could produce non-redundant complete test suites if appropriate test generation methods are applied;
- requires that these methods deal with partially specified, nondeterministic possibly unreduced FSMs and the reduction relation.
- relies on the testable representations which are not as tractable as composed machines.

6. COMPARING THE RESULTS OBTAINED WITH DIFFERENT STRATEGIES

We have considered in this paper the following grey-box-based testing strategies:

- Test derivation from the constructed composed machine based on the black-box representation of the given system;
- Test derivation based on an ad hoc choice of a subset of transitions to be checked in the composed machine;

- Test derivation based on the construction of a fault function;
- Test derivation from a testable behavior representation of the embedded component.

The order in which these strategies are placed reflects the achieved refinement of the fault domain. The fault domain for the strategy based on construction of the testable behavior is the smallest domain. The same example was used throughout this paper to illustrate the results obtained with different strategies. The results are summarized in Table 3. The last row gives the length of the test suites which are used for testing the component machines in isolation.

One can see from this table that the refinement of the fault domain has a tendency (at least in this example) to decrease the length of a corresponding complete test suite, as expected.

Table 3: Summary of the results of test derivation.

	Faults in		
Strategy	head component	tail component	head or tail component
black-box testing	63	63	63
ad hoc	44	52	59
fault function	37	42	55
testable representation	22	40	51
isolated testing	11	27	38

7. CONCLUSION

We have considered in this paper the problem of testing an implementation treated as a grey-box, assuming that the specification structure is preserved in all potential implementations, and that at most one component is faulty. We have presented and compared several possible testing strategies which rely on these assumptions in various manners. All these strategies have at least one common feature: they transfer in one way or another the problem of grey-box testing into the realm of black-box testing by finding a suitable black-box representation and an appropriate fault domain for which there already exists certain formal methods for test derivation. At the moment, every presented strategy seems to have its own suitability, although it should be mentioned that the strategy based on a constructed testable behavior of the component under test is capable of providing the complete characterization of detectable and undetectable faults, as well as the shortest complete test suite.

The basic techniques needed for constructing testable behavior have been presented in this paper for a simple system of communicating deterministic FSMs; however, they can be generalized to cover more complex systems of even nondeterministic machines. By comparing the testable representation of a component with the original one, it is possible to assess how its testability changes when it is embedded within the given system [Petr94]. The presented results give us some insight on design of "easily testable" systems; they could also be used to locate the points of observation and control. It is believed that the results presented in this paper for the FSM model can be applied to the LTS model as well, following a semantics-dependent transformation of an LTS into a corresponding FSM as it has been recently proposed in [Petr93a].

At the same time, having a good characterization of a testable behavior does not necessarily mean that there is a good algorithm. A lot of work remains to develop efficient methods for deriving tests from the nondeterministic FSM representation of the testable behavior. The grey-box testing requires efficient test derivation methods for various types of FSMs, especially for those which are nondeterministic and partially specified. The universality of the completeness assumption often implied by a number of existing methods seems to have been overestimated, and as a result there are not yet many test derivation methods for partially specified systems.

Acknowledgments. This work was partly supported by the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols, Université de Montréal, NSERC grant N=B020629188, "Analyse et conception des protocoles diagnosticables et modifiables", and by the Russian Found for Fundamental Research. The authors would like to thank G. v. Bochmann for stimulating discussions and S. A. Ezust for comments.

REFERENCES

[Brzo92] 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.

[Cast93] R. Castanet, O. Kone, "Deriving Coordinated Testers for Interoperability", IFIP Transactions, Protocol Test Systems, VI, (the Proceedings of IFIP TC6 Fifth International Workshop on Protocol Test Systems, 1993), Ed. by O. Rafiq, 1994, North-Holland, pp.331-345.

[Chow78] 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.

[Dam91] H. v. Dam, H. Klooster, E. Kwast, "Test Derivation for Standardized Test Methods", IFIP Transactions, Protocol Test Systems, IV (the Proceedings of IFIP TC6 Fourth International Workshop on Protocol Test Systems, 1991), Ed. by J. Kroon, R. J. Heijink and E. Brinksma, 1992, pp.69-82.

[Das75] P. Das, D. E. Farmer, "Fault-Detection Experiments for Parallel-Decomposable Sequential Machines", IEEE Transactions on Computers, Vol. C-24, No.11, 1975, pp.1104-1109.

[Deva89] S. Devadas, "Approaches to Multi-level Sequential Logic Synthesis", In Proceedings of the 26th Design Automation Conference, Las Vegas, 1989, pp.270-276.

[Drir93] K. Drira, P. Azema, B. Soulas, A.M. Chemali, "Testability of a Communicating System through an Environment", In Proc. of TAPSOFT'93.

[Fuji91] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, A. Ghedamsi, "Test Selection Based on Finite State Models", IEEE Transactions on Software Engineering, Vol. SE-17, No.6, 1991, pp.591-603.

[Gill62] A. Gill, Introduction to the Theory of Finite-State Machines, McGraw-Hill, 1962, 207p.

[Hopc79] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley Publishing Company, Inc., 1979, 418p.

[IS9646] ISO, OSI Conformance Testing Methodology and Framework, International Standard IS9646.

[Kim72] J. Kim, M. Newborn, "The Simplification of Sequential Machines with Input Restrictions", IEEE Transactions on Computers, Vol. C-21, No.12, 1972, pp.1440-1443.

[Lee93] D. Lee, K. Sabnani, D. Krispot, S. Paul, M. Uyar, "Conformance Testing of Protocols Specified as Communicating FSMs", INFOCOM'93, pp.115-127.

[Luo94] G. Luo, A. Petrenko, G. v. Bochmann, "Generating Tests for Communication Software Modeled by Partially-Specified Finite State Machines", IEEE/ACM Transactions on Networking, (to appear in 1994).

[Luo94a] G. Luo, A. Petrenko, G. v. Bochmann, "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.

[Morr93] D. Morris, B. Tamm, Concise Encyclopedia of Software Engineering, Pergamon Press, 1993, 400p.

[Petr91] A. Petrenko, "Checking Experiments with Protocol Machines", IFIP Transactions, Protocol Test 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.

[Petr92] A. Petrenko, N. Yevtushenko, "Test Suite Generation for a FSM with a Given Type of Implementation Errors", IFIP Transactions, Protocol Specification, Testing, and Verification, XII (the Proceedings of IFIP TC6 12th International Symposium on Protocol Specification, Testing, and Verification, 1992), Ed. by R.J. Linn. Jr. and M.U. Uyar, 1992, North-Holland, pp.229-243.

[Petr93a] A. Petrenko, G. v. Bochmann, R. Dssouli, "Conformance Relations and Test Derivation", IFIP Transactions, Protocol Test Systems, VI, (the Proceedings of IFIP TC6 Fifth International Workshop on Protocol Test Systems, 1993), Ed. by O. Rafiq, 1994, North-Holland, pp.157-178.

[Petr93b] A. Petrenko, N. Yevtushenko, A. Lebedev, A. Das, "Nondeterministic State Machines in Protocol Conformance Testing", IFIP Transactions, Protocol Test Systems, VI, (the Proceedings of IFIP TC6 Fifth International Workshop on Protocol Test Systems, 1993), Ed. by O. Rafiq, 1994, North-Holland, pp.363-378.

[Petr94] A. Petrenko, N. Yevtushenko, R. Dssouli, "Grey-Box FSM-based Testing Strategies", Department Publication 911, Université de Montréal, 1994, 22p.

[Rho91] J.-K. Rho, G. Hachtel, F. Somenzi, "Don't Care Sequences and the Optimization of Interacting Finite State Machines", Proceedings of the IEEE International Conference on Computer Aided Design, Santa Clara, CA, 1991, pp.414-421.

[Sidh89] D. P. Sidhu, T. K. Leung, "Formal Methods for Protocol Testing: A Detailed Study", IEEE Transactions on Software Engineering, Vol. SE-15, No.4, 1989, pp.413-426.

[Star72] P. H. Starke, Abstract Automata, North-Holland/American Elsevier, 1972, 419p.

[Vasi73] M. P. Vasilevski, "Failure Diagnosis of Automata", Cybernetics, Plenum Publishing Corporation, New York, No.4, 1973, pp.653-665.

[Vuon89] S. T. Vuong, W.W.L. Chan, and M. R. Ito, "The UIOv-method for Protocol Test Sequence Generation", IFIP Transactions, Protocol Test Systems, II, Proceedings of IFIP TC6 Second International Workshop on Protocol Test Systems, 1989, Ed. by J. de Meer, L. Machert and W. Effelsberg, pp.161-175.

[Yevt89] N. Yevtushenko and A. Petrenko, "Fault-Detection Capability of Multiple Experiments", Automatic Control and Computer Sciences, Allerton Press, Inc., New York, Vol.23, No.3, 1989, pp.7-11.

[Yevt90] N. Yevtushenko, A. Petrenko, "Method of Constructing a Test Experiment for an Arbitrary Deterministic Automaton", Automatic Control and Computer Sciences, Allerton Press, Inc., New York, Vol.24, No.5, 1990, pp.65-68.

[Yevt91] N. Yevtushenko, A. Lebedev, A. Petrenko, "On the Checking Experiments with Nondeterministic Automata", Automatic Control and Computer Sciences, Allerton Press, Inc., New York, Vol.25, No.6, 1991, pp.81-85.