Test suite minimization for testing in context

Nina Yevtushenko

Tomsk State University

36, Lenin str., Tomsk, 634050, Russia

Yevtushenko.RFF@elefot.tsu.tomsk.su

Ana Cavalli, Luiz Lima Jr.¹

National Institute of Telecommunications

9, rue Charles Fourier, 91011 Evry Cedex, France

{Ana.Cavalli,lima}@hugo.int-evry.fr

Abstract

In this paper, the problem of test suite minimization for testing in context is studied and some results are proposed. The test architecture considered is general enough to allow the context and the component to have external inputs and outputs. Using a given fault model for testing in context, conditions are provided to detect all the redundant transitions that do not need to be tested. A complete test suite for the fault model is obtained and a method to select a minimal subset of this test suite with the same fault detection power is then proposed. Furthermore, the proposed method can be used to reduce a test suite produced by human experts while preserving its fault detection power.

Keywords

Test derivation, test in context, embedded testing, test of communicating FSMs.

1 INTRODUCTION

As the complexity of communication protocols increases, there is a strong need for systematic methods for test derivation with guaranteed fault coverage based on formal description techniques. Formal description techniques for communication protocols usually use a system of communicating finite state machines (FSMs) as their

1. Work funded by CNPq.

underlying model. The ‘black-box’ testing of a system can be performed by constructing the composite machine that describes a behaviour of a system at points that are accessible for testing and then applying test suite derivation methods for FSMs [Vasilevsky 73][Chow 78][Vuong 89][Fujiwara 91][Petrenko 96-1][Yannakakis 95]. However, this approach provides very long tests due to a huge number of states of the composite machine.

If only some components of the system need testing, the problem of testing an embedded component machine (or ‘gray-box’ testing [IS-9646 91] or testing in context [Petrenko 96-2]) appears. In this case, a specification system is represented as a composition of two FSMs. One of these machines, called *component*, is the composite FSM of all machines which need testing. The other machine, called *context*, is the composite machine of all other machines that are assumed to be faulty-free. If the set of Points of Control and Observation (PCOs) of a system contains all the access points to the component (i.e. there are no internal signals exchanged between the component and the context), then testing in context reduces to ‘black-box’ testing in isolation and a number of methods can be called for deriving a test suite with guaranteed fault coverage. Otherwise, ‘black-box’ testing at the PCOs will (unnecessarily) test the context as well, that is assumed to be faulty-free. The issue has been studied in a number of papers [Lima 97][Petrenko 97][Petrenko 96-2][Huang 96][Lee 96][Kim 95][PT 96]. Nevertheless, as far as we know, the application of embedded testing methods in precise fields has not been greatly explored [Lima 98].

Not all transitions of the system come from the combination of transitions of the context and the component. Some concern uniquely the context and thus, do not need testing. Various methods for selecting such transitions have been proposed. Based on the so-called fault function, the authors of paper [Petrenko 96-2] suggest a method for translation of component’s transfer and output faults into faults of the composite FSM of the specification and call a method elaborated in [Petrenko 92] for complete test suite derivation. Another approach is based on a coverage of component’s transitions by performing a random walk of the specification composite FSM [Lee 96] but fault detection power of the method is unknown. Paper [Lima 97] is devoted to determining a certain part of the composite machine comprising transitions should be tested in the case when any fault of the component does not increase the number of states of the specification system. In the paper, transitions which the component does not affect are called redundant transitions while other transitions are called suspicious transitions. If the context is faulty-free then an external sequence traversing only redundant transitions is superfluous and can be deleted from the test suite without loss of its completeness. A method for determining such transitions is proposed, and sufficient conditions for deleting superfluous test cases are established. The authors notice, nevertheless, that a more rigorous analysis is necessary to elaborate a method for checking suspicious transitions. This paper continues this work.

The paper is structured as follows. After the preliminaries of Section 2, Section 3 explains how to deal with transitions concerning only the faultless context

(redundant transitions) in the machine representing the system's global behaviour. In Section 4, given a set of external input sequences (from Section 3), we propose a method for derivation of a regular language describing a part of the component's behaviour that can be tested with the set and a complete test suite minimization technique when the obtained regular set is finite. We conclude with a discussion on the future work in Section 5.

2 PRELIMINARIES

2.1 Finite State Machines

A Finite State Machine (FSM) (often simply called 'a machine' throughout this paper) is an initialized completely specified deterministic Mealy machine which can be formally defined as follows. A FSM A is a 6-tuple $(S, X, Y, \delta, \lambda, s_0)$ where S is a finite set of states with s_0 as the initial state; X - a finite nonempty set of input symbols; Y - a finite nonempty set of output symbols, δ and λ are the next state and output functions $\delta: S \times X \rightarrow S$ and $\lambda: S \times X \rightarrow Y$. In usual way, functions δ and λ are extended to functions on the set $S \times X^*$ where X^* is the set of all finite input sequences including the empty sequence ϵ . The FSM A is said to be *connected* if each state of A is reachable from the initial state, i.e. for each state $s \in S$ there exists an input sequence $\alpha \in X^*$ such that $\delta(s_0, \alpha) = s$.

Two states s_i and s_j are said to be *distinguishable* states of A if there exists an input sequence $\alpha \in X^*$ such that the FSM A produces different output sequences at the states s_i and s_j to the α ; otherwise, states s_i and s_j are said to be *equivalent*. An FSM with pairwise distinguishable states is called a *minimal* FSM.

Given sequences $\alpha = x_1 \dots x_k \in X^*$ and $\beta = y_1 \dots y_k \in Y^*$, the sequence $x_1 y_1 \dots x_k y_k$ is called a trace of the FSM A if $\beta = \lambda(s_0, \alpha)$. For a sequence σ over alphabets X and Y , $X \cap Y = \emptyset$, the X -projection of σ is obtained by deleting from σ all symbols of the set Y .

Given an FSM $B = (T, X, Y, \psi, \phi, t_0)$, B is said to be *equivalent to* A , written $B \equiv A$, if for any sequence $\alpha \in X^*$, it holds that $\lambda(s_0, \alpha) = \phi(t_0, \alpha)$, otherwise B is distinguishable from A , written $B \neq A$. In other words, B is equivalent to A if A and B exhibit the same behaviour under all input sequences. Protocol conformance testing is often formalized as FSMs equivalence problem.

2.2 Fault models

A fault model is a triple $\langle A, \sim, \mathfrak{R} \rangle$ [Petrenko 96-2], where A is a reference FSM, $\mathfrak{R}$, usually called a *fault domain*, is the set of all (possibly faulty) implementation FSMs defined over the same input alphabet as the reference machine, and $\sim$ is a conformance relation. If the reference FSM A is a deterministic minimal FSM with n states, $\mathfrak{R}_n$ is a set of all deterministic FSMs over the same input alphabet as A with at most n states and $\sim$ is the equivalence relation $\equiv$ then the fault model $\langle A, \equiv, \mathfrak{R}_n \rangle$ is a classical 'black-box' model. A complete test suite TS w.r.t. the fault model $\langle A, \equiv, \mathfrak{R}_n \rangle$ is a finite set of finite input sequences of the reference FSM A such that for any

FSM $B \in \mathcal{R}_n$, if $B \neq A$ then there exists input sequence $\alpha \in TS$ such that A and B produce different output sequences to α . There exists a number of methods for complete test suite derivation w.r.t. this model. Below we sketch the **Wp**-method [Fujiwara 91] for a complete test derivation.

Let $A = (S, X, Y, \delta, \lambda, s_0)$ be a minimal connected FSM with n states. A subset $V \subseteq X^*$ comprising the empty sequence ϵ is called a *state cover set* if for each state $s \in S$, there exists a sequence $\alpha \in V$ such that $\delta(s_0, \alpha) = s$. We further denote VX the set obtained by concatenating each sequence $\alpha \in V$ with each input $x \in X$. The set VX is usually called a *transition cover set* since each transition of the FSM A is traversed with an appropriate sequence of the set VX .

Given state $s \in S$ of the FSM A , a finite set W_s of finite input sequences is called a *state identifier* of the state s if for each state $p \in S$, $p \neq s$, there exists an input sequence $\alpha \in W_s$ such that $\lambda(p, \alpha) \neq \lambda(s, \alpha)$. An input sequence α is said to be a *distinguishing sequence* of the FSM A if the output responses of the FSM A at different states to α are different. When the FSM A has a distinguishing sequence α then the set $\{\alpha\}$ is a state identifier of any state of A . When a state identifier W_s is fixed for each state $s \in S$ a procedure for derivation of a complete test suite w.r.t. the fault model $\langle A, \equiv, \mathcal{R}_n \rangle$ comprises two phases.

In the former phase, the part T_1 of the test suite is derived. The T_1 checks whether an implementation FSM has exactly n states and each state identifier preserves its feature. This part of a test suite is obtained by concatenating each sequence $\alpha \in V$ with each sequence of the union of state identifiers W_s over all states $s \in S$. If the implementation and reference FSMs have one and the same output sequence to each sequence of T_1 then any state of the reference FSM has a corresponding state in the implementation FSM with the same state identifier.

The part T_2 of a test suite derived in the latter phase, checks whether every transition of the implementation FSM is correctly implemented. The T_2 is obtained by concatenating each sequence $\alpha \in VX$ with each sequence of the state

Figure 1 FSM RS.

identifier of the state $\delta(s, \alpha)$ where α takes the FSM A from the initial state. Merging the T_1 and T_2 we obtain a complete test suite $TS = T_1 \cup T_2$ w.r.t. the fault model $\langle A, \equiv, \mathcal{R}_n \rangle$. For the sake of simplicity, each sequence that is a prefix of another sequence can be deleted from the test suite TS .

Example 1. Consider an FSM RS in Figure 1 ($a1$ is the initial state). An input sequence x_2 is a distinguishing sequence of the RS. We select the state cover set $V = \{\epsilon, x_1\}$. Then $T_1 = \{x_2, x_1x_2\}$ while the transition cover set $VX = \{x_1, x_2, x_3, x_1x_1, x_1x_2, x_1x_3\}$. Concatenating each sequence of VX with x_2 we obtain $T_2 = \{x_1x_2, x_2x_2, x_3x_2, x_1x_1x_2, x_1x_2x_2, x_1x_3x_2\}$. After deleting from $T_1 \cup T_2$ all prefixes of other sequences the set $TS = \{x_2x_2, x_3x_2, x_1x_1x_2, x_1x_2x_2, x_1x_3x_2\}$ is obtained that is a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathcal{R}_2 \rangle$. The tree of the test suite is shown in Figure 2. The test suite comprises five test cases of total length 18. $\square$

Figure 2 The tree of a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathcal{R}_2 \rangle$. States that should be identified are depicted in dashed lines.

2.3 Composition of FSMs

We consider a composition as a system of two component FSMs, as shown in Figure 3. For the sake of simplicity, we consider pairwise disjoint sets X , Y , U and Z . The system at hand has a single message in transit, i.e. the environment submits the next external input to the system only when it has produced an external output to the previous input. This accounts for synchronous communication between the environment and the system. Notice, however, that asynchronous communication through bounded input queues can be simulated by explicitly introducing new contexts representing the behaviour of the bounded queue in the system [Tretmans 92][Phalippou 94]. The input queues must be bounded, so to assure that the corresponding FSM exists (i.e. it is really finite).

Under these assumptions, when there are no live-locks in the composition, we can derive the composite machine of FSMs C and $Comp$ using, for instance, the algorithm described in [Lima 97] in order to avoid generating the Cartesian product $C \times Comp$ (which is often huge).

Example 2. Consider the two machines C and $Comp$ shown in Figure 4a and Figure 4b. Considering states a and l the initial states, the composite machine $RS = C \bullet Comp$ is the FSM of Figure 1. $\square$

We now demonstrate that the composition of Figure 3 is general enough to discuss the problems related to testing a component machine. We claim that the gen-

Figure 3 Composition of two FSMs: the context C and the component $Comp$.

Figure 4 The context $C(a)$ and the component $Comp$ (b).

eral model of Figure 5 can be transformed into that in Figure 3 with the component and composite FSM being isomorphic to those of the original composition.

If the external actions X' and Y' are redirected through the context along with the replacement of the external alphabets X' and Y' by the internal alphabets U' and Z' (Figure 6), we obtain an equivalent (isomorphic) system due to one-to-one correspondence. For each input $x \in X'$, the context C at any state produces a corresponding internal output $u \in U'$ while for each internal input $z \in Z'$ the context produces the corresponding $y \in Y'$. $Comp$ is isomorphic to $EComp$ w.r.t. the one-to-one correspondence between X' and U' and Y' and Z' while the composite FSMs of the compositions coincide.

Figure 5 General composition of two FSMs. The context C' and the component $EComp$ have external inputs and outputs.

Figure 6 The transformed composition of two FSMs, the context C and the component $Comp$.

2.4 Fault model for testing in context

Many compound systems are formally specified as a system of interacting FSMs. If only some components of a system need testing we face the problem of testing an embedded component or testing in context. In this case, a system under test can be represented as the composition of two FSMs. As stated in Section 1, one of these machines, called the *component*, is the composite FSM of all component machines which need testing while another machine, called the *context*, is the composite machine of all other component machines that are assumed to be faulty-free. As demonstrated above, we can use the composition in Figure 3 to discuss problems related to testing in context.

We further suppose that the behaviour of a specification system is described by a deterministic FSM represented in the tester by a test suite. The tester executes test cases against an implementation system usually called a *System Under Test (SUT)*, and checks the results on receipt of a response from the SUT. In this paper, we suppose that the SUT's behaviour can be represented by a deterministic composite machine. If the output response of the SUT to a proper test case differs from that of the specification system, then the tester produces the verdict '**fail**'; in this case, the SUT is a nonconforming implementation of the specification system. Otherwise, the tester produces the verdict '**pass**' to the test case. If the verdict '**pass**' is produced to every test case of the test suite then we conclude that the SUT is a conforming implementation of the specification system.

Let a specification system be given as the composition of two interacting FSMs: the context C and the component $Comp$. Let this composition be represented by the composite machine RS with n states. An implementation system is also the composition of two interacting machines: the context C (a faultless implementation of the specification context C) and an implementation component machine Imp . Let the implementation system's behaviour be represented by the composite machine IM with at most n states. If $\mathfrak{R}_{n,C}$ is the set of all composite machines of such implementation systems, then we are required to derive a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$, i.e. a set of input sequences of RS such that for each implementation system of the domain that exhibits a different external behaviour from that of RS , there exists at least one test case when the tester produces the verdict '**fail**'.

As mentioned above, regular methods exist for the derivation of a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_n \rangle$. The test suite is also complete w.r.t. $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$ because $\mathfrak{R}_{n,C} \subseteq \mathfrak{R}_n$. The test suite checks the component as well, but it also checks (unnecessarily) the context that is assumed to be faulty-free; thus, it can be reduced in particular cases. In the paper [Lima 97], some sufficient conditions for reducing the test suite are established. These conditions are briefly summarized in the following section (Section 3.1).

3 DEALING WITH REDUNDANT TRANSITIONS

3.1 Reducing a test suite

Each transition of the composite machine RS comes from a combination of transitions of the two modules, the context and the component, but there may be transitions in the composite machine which are not affected by the component. In paper [Lima 97], such transitions are called *redundant* transitions; otherwise, a transition of the RS is called *suspicious*. If the context is assumed to be faulty-free then external sequences traversing only redundant transitions are superfluous and can be deleted from the test suite without loss of its completeness. A method for determining such transitions is proposed and sufficient conditions for deleting superfluous test cases are established. These conditions can be formalized as follows.

Proposition 3.1. Given the reference composite machine RS , let β be an external input sequence only traversing redundant transitions of the RS . The output sequence of an implementation system to β coincides with that of FSM RS .

Corollary 3.2. Given a complete test suite TS w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$, let $\beta \in TS$ be a sequence only traversing redundant transitions of the RS . Then $TS \setminus \{\beta\}$ is also a complete test suite w.r.t. the fault model.

Example 3. Consider sequences x_2x_2 and x_3x_2 of the complete test suite TS w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_2 \rangle$ obtained in Section 2.2. The sequences only traverse redundant transitions $(a1, x_2)$, $(a1, x_2)$ and $(a1, x_3)$, $(a1, x_2)$ of the composite FSM RS . Thus, $TS = \{x_1x_1x_2, x_1x_2x_2, x_1x_3x_2\}$ is a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{2,C} \rangle$. It has three sequences of total length 12. All sequences of the set TS traverse a suspicious transition. $\square$

Nevertheless, it was pointed out that a more rigorous analysis is necessary in order to elaborate a method for checking suspicious transitions.

3.2 Redundant transitions

It may well happen that a redundant transition of the specification system becomes suspicious due to an unexpected interaction in the implementation system that brings the context to a wrong state. Consider, for instance, the system of two FSMs: context D (Figure 7a) and component $Spec$ (Figure 7b).

Figure 7 The context D (a) and the component $Spec$ (b).

The composite FSM is shown in Figure 8.

We use input sequence x to reach state $b2$ from the initial state $a1$ and the same sequence x as a distinguishing sequence. Thus, $T_1 = \{x, xx\}$ coincides with $VX = \{x, xx\}$ and $T_2 = \{xx, xxx\}$, i.e. $TS = \{xxx\}$ is a complete test suite w.r.t. the fault model $\langle D \cdot Spec, \equiv, \mathfrak{R}_2 \rangle$. Note that the transition from the state $b2$ under input x is a redundant transition. If we do not check this transition, then $TS' = \{xx\}$. However, TS' is not complete w.r.t. the fault model $\langle D \cdot Spec, \equiv, \mathfrak{R}_{2,C} \rangle$.

Figure 8 The composite FSM $D \cdot Spec$.

Assume that the implementation component FSM Imp depicted in Figure 9a replaces $Spec$. The composite FSM $D \cdot Imp$ is shown in Figure 9b. By direct inspection, one can assure that the input sequence xxx is the shortest sequence distinguishing nonequivalent FSMs $D \cdot Spec$ and $D \cdot Imp$.

The reason is that an implementation component machine, being an arbitrary FSM over alphabets U and Z , can induce unexpected internal interactions in the implementation system. These unexpected internal interactions bring the implemen-

Figure 9 The implementation component FSM *Imp* (a) and the implementation composite FSM *D•Imp* (b).

tation system to a wrong state while producing the expected output. These kinds of faults are known as *latent faults* [Petrenko 96-2] and can only be detected when longer external test cases are submitted. Therefore, some effort must be put into characterizing such cases, so to improve the proposed solution.

Given an external input sequence $\alpha \in X^*$ of the FSM *RS* and an internal trace over alphabets U and Z , the trace is said to be unexpected w.r.t. α if the trace may be induced in an implementation system not coinciding with that of the specification system when α is submitted. Unexpected internal traces represent alternative paths that a system may take due to faults in the implementation component machine.

Using the concept of unexpected internal traces in the previous example, we observe that a latent fault appears because:

1. there exist unexpected internal traces uz_2 and uz_3 w.r.t. the external input x ; and
2. one of these unexpected internal traces, namely uz_3 , produces the expected external output y_1 .

Generally speaking, given a redundant transition of the FSM *RS* at state s , the latent fault may occur under the following conditions:

1. the external input sequence $\alpha \in VX$ that takes the *RS* from its initial state to state s traverses suspicious transitions; and
2. at least one of the unexpected internal traces w.r.t. α results in the expected output sequence.

Based on this observation we conclude that a redundant transition becomes suspicious if conditions 1 and 2 above apply, and then the tail state of such transition must be checked as well.

On the other hand, if for some test case of a given test suite, none of the unexpected internal traces w.r.t. the case result is the expected external output sequence, then the output sequence of an implementation to the test case coincides with that of the specification system. In this case, the test case is superfluous and can be deleted from the test suite without loss of its completeness. When at least one of the unexpected internal cases w.r.t. the test case results in an unexpected external output sequence, other test case should be examined to recognize those that are superfluous.

4 TEST SUITE MINIMIZATION

4.1 Detectable internal traces

The example of Section 3.2 clearly demonstrates that the fault detection power of an external input sequence α in the given context can be characterized by the set of all unexpected internal traces such that, if an implementation component machine comprises such a trace, then the specification and implementation systems produce different output sequences to α . We call such internal traces *detectable with the external input sequence α* . If an implementation component FSM has no traces detectable with α , then the specification and implementation systems produce the same output sequence to α . The formal definition of such internal traces is given in [Petrenko 97] where they are called forbidden internal traces w.r.t. α . As the set of traces of an FSM is a prefix-closed set, an extension of a trace which is detectable with an external input sequence is also detectable with this sequence. Conversely, a set of internal traces that are detectable with an external input sequence includes that for its prefix.

Given a set T of external input sequences, an internal trace is said to be *detectable with the set T* if it is detectable with at least one sequence of the set. Due to the definition of detectable traces, the following statement holds.

Proposition 4.1. Given the composite FSM RS of the specification system with the context C and a complete test suite TS w.r.t. fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$, let P be a subset of TS . If all internal traces detectable with TS are detectable with P then P also is a complete test suite w.r.t. the fault model.

Due to Proposition 4.1, to compare the fault detection power of two test suites we need a procedure for characterizing internal traces that are detectable with a given external input sequence. We represent such traces as sequences recognized by a designated final state 'fail' of an appropriate acceptor or recognizer [Hopcroft 79]. Our first step is to describe all possible traces that may be induced in an implementation system when the external sequence is submitted. Because a component FSM is completely embedded within the context (see Section 2.3), these traces can be described as traces of the context LTS LC obtained by unfolding each atomic transition *input/output* in the context FSM C . The context LTS LC for the context C in Figure 4a is shown in Figure 10.

Given an external input $x \in X$, we construct the acceptor $LC(x)$. States of the acceptor are states of LC with the initial state s_0 . The initial and final states of the acceptor are special states that cannot be merged with other states with the same names called *intermediate states*, while it is possible to merge two such intermediate states. There is a transition labelled with x from the initial state to state s if x takes the LC from the initial state s_0 to the state s . For two intermediate states p and r , there is a transition labelled with action $a \in U \cup Z$ if there is a transition labelled with a from state p to state r in the LTS LC . There is a transition labelled with $y \in Y$ from the intermediate state p to the final state s if there exists an outgoing transition

at the state p to state s labelled with y in the LTS LC . The connected part of the acceptor comprising the initial state is denoted $LC(x)$.

Figure 10 The context LTS LC representing all the traces of the context C .

$LC(x_2)$ with s as the initial state. The construction of the acceptor $LC(x_1...x_k)$ implies the following statement.

Proposition 4.2. Given the acceptor $LC(x_1...x_k)$, let IS be the implementation system of a component over alphabets U and Z within context C . 1) If σ is a trace of the system when the external input sequence $x_1...x_k$ is submitted, then the acceptor $LC(x_1...x_k)$ has a path from the initial state to a final state labelled with the sequence σ . 2) If an $(U \cup Z)$ -projection of sequence σ labelled a path from the initial to a final state is a trace of the implementation component, then the output sequence of the implementation system to $x_1...x_k$ is the Y -projection of the sequence σ .

Example 4. Consider an implementation system with the context of Figure 4a and an input sequence $x_1x_2x_2$. When the external input x_1 is applied to the context at the initial state a the context produces the internal output u and enters the state b . Its next action depends on an output produced by a component machine to the input u . If the component produces z_1 then the context remains at the state b producing y_1 while the context enters the state a producing y_1 if the component has the output response z_2 to u . In both cases, the context produces an external output y_1 to x_1 and the next external input x_2 can be applied to the context. Since we are interested in all the traces that may be induced when the external input sequence is submitted to the context we do not merge states with the same names separated with external actions. The procedure of constructing the acceptor $LC(x_1x_2x_2)$ is illustrated in Figure 11. $\square$

The next step is to transform the acceptor, so that the set of all internal traces detectable with the given external input sequence would be represented by the set of $(U \cup Z)$ -projections of the sequences recognized by a designated state 'fail.'

Procedure 4.1. Derivation of the regular expression for forbidden traces w.r.t. a given external input sequence.

Given a final state s of the acceptor $LC(x)$, the set of all sequences labelling the paths from the initial to the final state s is a regular set recognized by the final state s [Hopcroft 79]. Due to construction of the acceptor, the regular expression specifying the set can be written as xfy where f comprises only internal actions of the set $U \cup Z$. Thus, the set of the $(U \cup Z)$ -projections of the regular set recognized by a final state also is a regular set specified by a proper regular expression.

To construct the acceptor $LC(x_1...x_k)$ we construct at each terminal state s of the acceptor $LC(x_1)$ the acceptor

Inputs: The composite machine RS of the specification system with a context C and an external input sequence α .

Output: A regular expression E_α for internal traces detectable with α .

Step 1. Derive the context LTS LC unfolding each atomic transition *input/output* in the context FSM C and construct the acceptor $LC(\alpha)$.

Step 2. For each path of the acceptor $LC(\alpha)$ from the initial state to a final state such that the $(X \cup Y)$ -projection of the sequence labelled the path is not a trace of the specification composite FSM RS , replace the last state of the path with a designated deadlock state 'fail'.

Step 3. If for some transition labelled with external input $x \in X$ or with an internal action $z \in Z$, all the subsequent paths have a final state 'fail' then replace the final state of the transition with the 'fail' state.

Step 4. Derive a regular expression E_α for $(U \cup Z)$ -projections of sequences labelling all paths from the initial state to **FAIL** state.

The regular language obtained by Procedure 4.1, is called the *characterization fault detection set* $D(\alpha)$ of the sequence α which completely characterizes the set of nonconforming implementation systems that can be detected with the external input sequence α .

Proposition 4.3. Given a context C , a component specification $Comp$, a component implementation Imp , the composite machines RS (system specification) and IS (system implementation), and an external input sequence α , let E_α be a regular expression obtained by the Procedure 4.1. The external sequence α distinguishes FSMs RS from IS iff the language specified by the expression E_α comprises a trace of the component Imp of the implementation system.

Proof. First part. Let the language specified by the regular expression E comprise a trace β/γ of a component Imp . By construction of E_α , there is a path in the

Figure 11 The fragment of the acceptor $LC(x_1x_2x_2)$. The external projections of executed traces are underlined.

acceptor (obtained at Step 2 of Procedure 4.1) from the initial state to the ‘fail’ state labelled with a sequence such that: 1) its $(U \cup Z)$ - projection $\beta\beta'/\gamma\gamma'$ is a trace of the implementation component *Imp*; 2) its X - projection is α ; and 3) its Y - projection σ is different from the *RS*’s output sequence to α . The ‘fail’ state replaces an appropriate final state of the acceptor $LC(\alpha)$. Due to Proposition 4.2 (Part 2), the implementation system produces the output sequence σ when α is submitted to the system.

Second part. We now assume that the *IS* has the output sequence σ to α that is different from that of the *RS*. Then the acceptor obtained at Step 2 of Procedure 4.1 comprises the path from the initial state to the ‘fail’ state labelled with a trace induced in the implementation system when α is submitted to the system (Proposition 4.2, Part 1). Therefore, the X - projection of the sequence labelling the path is α , the Y - projection is σ and the $(U \cup Z)$ - projection is a trace β/γ of the component *Imp*. By construction, the language specified by regular expression E comprises a prefix to β/γ , i.e. a trace of the component *Imp*. $\square$

Corollary 4.4. Given an internal trace τ and an external input sequence α , τ is detectable with α iff the set $D(\alpha)$ contains a prefix of τ .

Given the set T of external input sequences, the union of the characterization fault detection sets over all sequences in the set T is called the *characterization fault detection set* $D(T)$ of T .

Example 5. Consider a sequence $x_1x_2x_2$ of the complete test suite $TS = \{x_1x_1x_2, x_1x_2x_2, x_1x_3x_2\}$ w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{2,C} \rangle$ obtained in Section 3 after deleting test cases traversing only redundant transitions. This sequence traverses suspicious transitions. We need now to determine the characterization fault detection set $D(x_1x_2)$ for its prefix x_1x_2 .

Consider in the acceptor the path labelled with the sequence $x_1uz_2y_1x_2y_1$ from the initial state. A system executes this sequence of actions when the context is combined with an implementation component machine producing at the initial state the output z_2 to the input u . Once the component of a system produces the output z_2 to the input u , the system will produce the output sequence y_1y_1 to the external input sequence x_1x_2 . This output does not coincide with the expected output y_1y_2 of the reference composite FSM (Figure 1) to the input x_1x_2 . Thus, the final state of the path is replaced by a ‘fail’ state.

Consider now the paths labelled with sequences $x_1uz_1y_1x_2uz_1uz_1y_1$ and $x_1uz_1y_1x_2uz_1uz_2y_1$. A system executes these sequences of actions when the context is combined with an implementation component machine producing at the initial state the output sequence $z_1z_1z_1$ or $z_1z_1z_2$ to the input sequence uuu . Thus, since the implementation component of a system at hand produces the output sequence $z_1z_1z_1$ (or $z_1z_1z_2$) to the input sequence uuu , the system produces an output sequence (y_1y_1) to the input sequence x_1x_2 that is different from the expected output sequence y_1y_2 of the reference FSM *RS* (Figure 1). The prefix $x_1uz_1y_1x_2uz_1$ of the sequences takes the acceptor from the initial state to a state where all the subsequent paths lead to a ‘fail’ state. In other words, if an implementation component machine has a trace uz_1uz_1 , then the system will always produce the unexpected output sequence y_1y_1 to the input sequence x_1x_2 , regardless of the output of the implementation component

to the next input u . That is why we replace the state where $x_1uz_1y_1x_2uz_1$ takes the acceptor to from the initial state by a 'fail' state. Therefore, $D(x_1x_2) = \{uz_1uz_1, uz_2\}$. By direct inspection of Figure 11, one can assure the set $D(x_1x_2x_2)$ coincides with $D(x_1x_2)$, i.e. the test case $x_1x_2x_2$ can be replaced by x_1x_2 without loss of completeness of the TS w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{2,C} \rangle$.

On the other hand, if the component of an implementation system has no traces of the set $D(x_1x_2)$, then even unexpected internal interactions in an implementation system result in the expected external output sequence y_1y_2 of the reference FSM RS to the external sequence x_1x_2 . In a similar way, we determine the sets $D(x_1x_1x_2) = \{uz_2uz_2, uz_2uz_1uz_1, uz_1uz_1\}$ and $D(x_1x_3x_2) = \{uz_2, uz_1uz_1\}$. Thus, the fault characterization set $D(TS) = \{uz_2uz_2, uz_2uz_1uz_1, uz_2, uz_1uz_1\}$ comprises a prefix of any internal trace detectable with the test suite TS . $\square$

Once the component of an implementation system has a trace belonging to $D(TS)$, there exists a test case in the TS to which the implementation and reference systems produce different output sequences. Conversely, if the component of an implementation system has no traces of the set $D(TS)$ then the implementation and reference systems produce the same output sequence to each test case of the TS . In other words, if the composite machine of the implementation system has at most two states then it is equivalent to the RS . That is to say that all components having no traces of the set $D(TS)$ that combined with the context possess the composite machine with at most two states are equivalent to the specification component $Comp$ in the context C . At the PCOs it is impossible to recognize which of them serves as the component of the system at hand.

4.2 Minimizing a test suite

Removing redundant transitions

Using the results of Section 3 and Section 4.1, we can reduce the procedure of derivation of a complete test suite w.r.t. the fault model $\langle A, \equiv, \mathfrak{R}_n \rangle$ (as explained in Section 2.2) to the procedure to generate a complete test suite w.r.t. the fault model $\langle A, \equiv, \mathfrak{R}_{n,C} \rangle$.

Procedure 4.2. Deriving a complete test suite w.r.t. the fault model $\langle A, \equiv, \mathfrak{R}_{n,C} \rangle$.

Input: The composite FSM RS of a specification system with n states that is minimal and connected; the state cover set of RS ; a transition cover set; and the set of state identifiers for all the states of RS .

Output: A complete test suite w.r.t. the fault model $\langle A, \equiv, \mathfrak{R}_{n,C} \rangle$.

Step 1. Concatenate each sequence of the state cover set with each sequence of the union of the state identifiers over all states of the FSM RS . Denote T_1 the set obtained. Let Q' be the subset of states of the context such that, for any state $q \in Q'$, there exists a sequence of the state cover set traversing only redundant transitions taking the context from the initial state to state q .

Step 2. Concatenate each sequence of the transition cover set traversing either a

suspicious transition or taking the FSM RS from the initial state to state (q, t) , ($q \notin Q'$) with each sequence of the identifier for the state to which the sequence takes the FSM RS from the initial state. Denote T_2 the set obtained and TS the union of T_1 and T_2 . Each sequence that is a prefix of another sequence can be deleted from the TS .

Comparing the above procedure with that in Section 2.2 we do not include in T_2 any subset obtained by concatenating a sequence of the transition cover set traversing only redundant transitions and taking the FSM RS from the initial state to the state (q, t) , where $q \in Q'$, with an appropriate state identifier, i.e. we do not test an unnecessary transition [Lima 97].

Proposition 4.5. Given the composite FSM RS of the specification system with n states, the set TS derived by Procedure 4.2 is a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$.

Proof. Let α be a sequence of the state cover set traversing only redundant transitions and taking the FSM RS from the initial state to state (q, t) , where $q \in Q'$. Moreover, let β be a sequence of the transition cover set traversing only redundant transitions and taking the FSM RS from the initial state to the state (q, t') . By construction, the set T_1 contains all sequences of the state identifier $W_{qt'}$ for state (q, t') . Each internal trace detectable with the sequence $\beta\gamma$, $\gamma \in W_{qt'}$ is detectable with $\alpha\gamma$ because the parts of both acceptors $LC(\beta\gamma)$ and $LC(\alpha\gamma)$ comprising internal traces start at the same initial state q of the context. Thus, the subset containing the concatenation of the subsequence β with each sequence of the set is superfluous in the complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$.

Deleting superfluous test cases

Suppose now that we derive a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$ using Procedure 4.2 (i.e. not including into the test suite a sequence traversing only redundant transitions - Corollary 3.2). There may exist a proper subset of the test suite such that any internal trace detectable with the test suite is detectable with the subset, i.e. some sequences of the test suite may still be superfluous. Due to Proposition 4.3, to compare a fault detection power of two external input sequences, it is sufficient to compare the corresponding regular sets (the comparison of arbitrary regular sets is out of the scope of this paper). Let us assume that the characterization fault detection set $D(TS)$ is finite for a given complete test suite TS w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$, as it often occurs in practical situations. In this case, the problem of determining a minimal subset of a complete test suite TS , that is also complete w.r.t. $\langle RS, \equiv, \mathfrak{R}_{n,C} \rangle$, is reduced to derivation of a minimal column coverage P of the boolean matrix B , i.e. a minimal row subset of P such that for each column there exists at least one row of P comprising 1s in this column. The rows of the matrix B correspond to sequences of the test suite TS while the columns correspond to internal traces of the characterization fault detection set $D(TS)$. Item b_{ij} is '1' iff

for the sequence $\alpha_i \in TS$, the characterization fault detection set $D(\alpha_i)$ contains a prefix of the internal trace $(\beta/\gamma)_j \in D(TS)$.

Example 6. The boolean matrix B for our working example is shown in Table 1. By direct inspection, one can assure that set $D(x_1x_2)$ contains the prefixes of all traces of the set $D(TS)$, i.e. all internal traces detectable with the test suite $TS = \{x_1x_1x_2, x_1x_2, x_1x_3x_2\}$ (obtained in Section 4.1) are detectable with the external input sequence x_1x_2 . Therefore, the subset $\{x_1x_2\}$ is a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_{2,C} \rangle$ being sufficient to detect an implementation system that possesses the composite machine with at most two states and is not equivalent to the reference composite FSM RS in Figure 1 when the context is faulty-free. One can compare this test suite (of total length 2) with a complete test suite w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_2 \rangle$ obtained in Section 2.2 that comprises five test cases of total length 18. $\square$

In a number of practical situations it is nearly impossible to derive the composite machine of the overall system due to its huge number of states. In this case, a test suite often is derived by a test engineer who is a high-level expert in the area. The obtained test suite checks important features of the protocol's implementations but it is difficult to estimate its fault coverage in the formal way. The above approach can be used to reduce the given test preserving its fault detection power.

Let a specification system be a composition of the context C and the component $Comp$ possessing the composite machine RS and let $\mathfrak{R}$ be a finite set of FSMs over the same input alphabet as RS . We denote $\mathfrak{R}_C$ the subset of $\mathfrak{R}$ comprising each machine of the set $\mathfrak{R}$ that is the composite machine of some implementation system with the same context. Given a set T of test cases that is complete w.r.t. the fault model $\langle RS, \equiv, \mathfrak{R}_C \rangle$, a subset P of the set T is said to have the same *fault detection power in the given context* if it is also a complete test suite w.r.t. that fault model.

TABLE 1. The boolean matrix B

	u			
	z ₂			
u	u	u		
z ₂	z ₁	z ₁		
u	u	u	u	
z ₂	z ₁	z ₁	z ₂	
$x_1x_1x_2$	1	1	1	0
x_1x_2	1	1	1	1
$x_1x_3x_2$	1	0	1	0

Given the context C and the set T of external test cases, we can use Procedure 4.1 to determine the characterization fault detection $D(T)$ and derive its subset P with the same fault detection power as a minimal coverage of an appropriate boolean matrix when $D(T)$ is finite.

5 FUTURE WORK AND CONCLUSION

Our future work is basically related to the generalization of the proposed approach. It can be easily generalized to the case when some transitions of the specification context and component are undefined while their implementations are assumed to be completely specified. In undefined situations, the implementation can produce an error message, for example, or have a loop labelled with the **NULL** output. In this case, conformance testing for protocols can be formalized as FSMs quasi-equivalence problem where a complete FSM B is said to be quasi-equivalent to (possibly partial) FSM A if A and B exhibit the same behaviour under all input sequences where a behaviour of A is specified. We can call methods for a complete test suite derivation w.r.t. the fault model $\langle A, \equiv_q, \mathcal{R}_n \rangle$ [Petrenko 96-1][Yannakakis 95] (where $\equiv_q$ is a quasi-equivalence relation) and use the method proposed in this paper to minimize the obtained test suite.

We also would like to generalize this approach to a system of communicating nondeterministic FSMs. As the authors of the paper [Lima 97] notice, Proposition 3.1 and Corollary 3.2 hold in this case. But now it is insufficient to keep in a test suite external sequences detecting each trace detectable w.r.t. the test suite. The subset of remaining test cases should also check whether the set of output responses of an IUT to an external input sequence α contains each output sequence of the reference composite FSM to α . The problem is closely related to the problem of equation solving [Merlin 83][Kim 72][Parrow 89][Watanabe 93], where we are required to derive the largest specification of the sub-module that combined with the given context is equivalent to the specification's composite FSM. To the best of our knowledge the problem is not solved to the general case; solutions only exist for particular cases.

In this paper we presented a test derivation strategy for testing in context. The system studied was composed of two communicating FSMs, the context and component, and the test architecture was generic, i.e. the context and the component may have external inputs and outputs. Using the given fault model for testing in context, conditions were provided to detect redundant transitions that did not need to be tested.

Furthermore, given an external input sequence, a regular set was derived such that a nonconforming implementation system could be detected with this input sequence if and only if its component comprised a trace of the set. Based on this approach a method was also presented to construct a complete test suite and to select a minimal subset of this test suite having the same fault detection power. On the other hand, the proposed method can be used to reduce a test suite given by human experts while preserving its fault detection power.

6 REFERENCES

- [Chow 78] T. S. Chow, *Test software design modelled by finite state machines*, IEEE Trans. SE-4, No. 3, pp. 178-187, 1978.
- [Fujiwara 91] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, A. Ghedamsi, *Test selection based on finite state models*, IEEE Trans., SE-17, No. 6, 1991, pp. 591-603.
- [Hopcroft 79] J. E. Hopcroft, J. D. Ulman, *Introduction to automata theory, languages and computation*, 1979, Addison-Wesley, NY.
- [Huang 96] S. Huang, D. Lee, M. Stauskas, *Validation-based test sequence generation for networks of extended finite state machines*, Proc. of the 1st Joint Intern. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols., and Protocol Specification, Testing, and Verification. 1996, pp. 403-418.
- [IS-9646 91] *Information Technology, Open Systems Interaction, Conformance Testing Methodology and Framework*, International Standard IS-9646. ISO, 1991.
- [Kim 72] J. Kim, M. Newborn, *The simplification of sequential machines with input restrictions*, IEEE Trans. on Computers, C-20, pp. 1440-1443, 1972.
- [Kim 95] M. C. Kim, S. T. Chanson, S. Yoo, *Design for testability of protocols based on formal specifications*, Proc. of the 8th Intern Workshop on Protocol Test Systems, 1995.
- [Lee 96] D. Lee, K. K. Sabnani, D. M. Kristol, S. Paul, *Conformance testing of protocols specified as communicating finite state machines - a guided random walk based approach*, IEEE Trans. on Communication, 1996, vol. 44, 5, pp. 631-640.
- [Lima 97] L. P. Lima, A. R. Cavalli, *A pragmatic approach to generating test sequences for embedded systems*, Proc. of the 10th Intern. Workshop on Protocol Test Systems, 1997.
- [Lima 98] L. P. Lima, A. R. Cavalli, *Application of embedded testing methods to service validation*, submitted to 2nd IEEE International Conference on Formal Engineering Methods, Brisbane, Australia, 1998.
- [Merlin 83] P. Merlin et al. *On the construction of submodule specifications and communication protocols*, ACM Trans. on Programming Languages and Systems. Vol. 5, No. 1, 83.
- [Parrow 89] J. Parrow, *Submodule construction as equation solving in CCS*, Theoretical Computer Science, 68, 1989, pp. 175-202.
- [Petrenko 97] A. Petrenko, N. Yevtushenko, *Testing faults in embedded components*, Proc. of the 10th Intern. Workshop on Protocol Test Systems, 1997.
- [Petrenko 96-1] A. Petrenko, N. Yevtushenko, G. v. Bochmann, *Testing deterministic implementations from nondeterministic FSM specifications*, Proc. of the 9th Intern Workshop on Protocol Test Systems, 1996.
- [Petrenko 96-2] A. Petrenko, N. Yevtushenko, G. v. Bochmann, *Fault models for testing in context*, Proc. of the 1st Joint Intern. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing, and Verification. 1996, pp. 125-140.
- [Petrenko 92] A. Petrenko, N. Yevtushenko, *Test suite generation for a given type of implementation errors*, Proc. of the Intern. Conf. on Protocol Specification, Testing, and Verification. 1992.
- [Phalippou 94] M. Phalippou, *"Relations d'implantation et hypothèses de test sur des automates à entrées et sorties"*, Ph.D. Thesis, Université de Bordeaux I, 1994.
- [PT 96] *Component testing for mobile and broadband telecommunications*, COIMBRA, COPERNICUS project Proposal, 1996.
- [Tretmans 92] J. Tretmans, *"A Formal Approach to Conformance Testing"*, Ph.D. Thesis, 1992.

- [Vasilevsky 73] *M. P. Vasilevsky, Failure diagnosis of automata. Cybernetics*, Plenum Publishing Corporation, NY, No. 4, 1973, pp. 653-665.
- [Vuong 89] *S. T. Vuong, W. W. L. Chan, M. R. Ito, The UIOv-method for protocol test sequence generation*, Proc. of IFIP TC6 Second Inter. Workshop on Protocol test Systems. 1989, pp. 161-175.
- [Watanabe 93] *Y. Watanabe, R. K. Brayton, The maximal set of permissible behaviours for FSM networks*. Proc. of the IEEE/ACM Intern. Conf. on Computer-Aided Design, 1993, pp. 316-320.
- [Yannakakis 95] *M. Yannakakis, D. Lee, Testing finite state machines: fault detection*, Journal of Computer and System Sciences, 1995, 50, pp. 209-227.

7 BIOGRAPHY

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

Luiz Paula Lima Jr. is currently a PhD student at INT (*Institut National des Télécommunications*), Evry, France and he received his MSC degree in 1994 at UNICAMP (State University of Campinas), Brazil. His current research interests include object-oriented distributed systems and platforms (ODP/CORBA) and testing methods for these architectures.

Ana Rosa Cavalli received the Doctorat d'État es Mathematics and Computer Science in 1984 from the University of Paris VII, Paris, France. From 1985 to 1990, she was a staff research member at the CNET (*Centre National d'Etudes des Télécommunications*), where she worked on software engineering and formal description techniques. Since 1990, she joined the INT (*Institut National des Télécommunications*) as professor. Her research interests include formal description techniques, validation of protocols and services, computing methodology and testing methods for distributed architectures.