Test suite minimization for embedded nondeterministic finite state machines

Nina Yevtushenko *Ana Cavalli*
Ricardo Anido

Relatório Técnico IC–99-11

Abril de 1999

Test suite minimization for embedded nondeterministic finite state machines

Nina Yevtushenko* Ana Cavalli † Ricardo Anido ‡

Abstract

This paper presents a method for minimizing test suites for embedded, nondeterministic Finite State Machines. The method preserves the fault coverage of the original test suite, and can be used in conjunction with any technique for generating test suites. The minimization is achieved by detecting and deleting redundant test cases in the test suite. The proposed method is an extension of the work presented in [YCL98].

1 Introduction

One important aspect of automatic test generation that has been investigated recently is the derivation of tests for a component embedded within a complex system. This problem is known as gray-box testing, embedded testing ([ISO91]) or testing in context ([PYvBD96]), and corresponds to a situation quite common in practice, where the system to be tested (the component) is part of a larger system, and can only be tested in the context of another part assumed to be fault-free (the context). A number of approaches have been developed for testing in context ([PYvBD96, PYvB96, LSKP96, LC97, LC98, PY97, YCL98]), some of which are heuristic and do not guarantee complete fault coverage ([LSKP96]). Other approaches deliver complete test suites with respect to various fault domains (i.e., w.r.t. sets of possible implementations of a component under test). Examples of the latter approach are methods which start with a test suite either derived by classical procedures such as W-, Wp, UIO, SC methods ([Vas73, Cho78, FvBK$^+$91, VCI89, YL95, LY96]) or given by human experts; the derived test suite is afterwards reduced without loss of its completeness.

In this paper we propose a method for minimizing a test suite for communicating nondeterministic Finite State Machines (FSMs) while maintaining its fault-coverage. We consider a specification system composed of two communicating FSMs. One of these machines, called component, is the machine that needs testing. The other machine, called context, describes the behavior of the part of the system which is assumed to be correctly implemented. Component, context or the overall system may be nondeterministic.

*Tomsk State University,36 Lenin av., 634050 Tomsk, Russia

†National Institute of Telecommunications, 9, rue Charles Fourier, 91011, Evry Cedex, France

‡University of Campinas, Cx Postal 6167 UNICAMP, 13083-970 Campinas SP, Brazil — Work supported by FAPESP and Pronex/Finep/SAI n. 76.97.1022.00

The method proposed extends the work presented in ([YCL98]), for deterministic FSMs, to nondeterministic FSMs. For each test sequence of a given test suite, we determine a regular set of internal traces that can be induced within an arbitrary implementation system. The regular sets thus obtained completely characterize the fault detection power of the test suite. We then present a procedure for determining a minimal proper subset of the test suite which accepts all nondeterministic behaviors accepted by the original test suite while maintaining its fault-detection power. The rest of the paper is structured as follows. Section 2 presents some basic notions. Section 3 analyzes the conditions under which a test case can be removed from a test suite without loss of its completeness. Section 4 describes how the problem of minimizing a given test suite can be reduced to determining a minimal column coverage of a Boolean matrix, and in Section 5 we present some conclusions.

2 Preliminaries

This section presents some basic notions on composition of FSMs and conformance testing.

2.1 Finite state machines

A Finite State Machine (FSM, often simply called a machine throughout this paper) is an initialized (possibly, nondeterministic) machine denoted by a 5-tuple $A = (S, X, Y, h, s_0)$, where S is a finite nonempty set of states with $s_0 \in S$ as the initial state, X is a finite nonempty set of inputs, Y is a finite nonempty set of outputs, and h is a behavior function, $h : S \times X \to P(S \times Y)$ where $P(S \times Y)$ is a set of all nonempty subsets of $S \times Y$. FSM A is *deterministic* if $\mid h(s,x) \mid = 1$ for all $(s,x) \in S \times X$. The machine is *observable* if for all $(s,x) \in S \times X$ and $y \in Y$ it holds that $\mid (s,y) \mid (s,y) \in h(s,x) \mid \leq 1$. In the usual way, the function h is extended to a function on the set $S \times X^*$ with results in the set $P(S \times Y*)$ where X^* is the set of all finite sequences in X containing the empty sequence ϵ. The function h has two projections h^s and h^y, usually called the next state function and the output function. FSM A is said to be *connected* if each state of A is reachable from the initial state, i.e. for each state s of A there exists an input sequence α such that $s \in h^s(s_0, \alpha)$.

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 at state $s \in S$ of FSM $A = (S, X, Y, h, s0)$ if $\beta \in h^2(s, \alpha)$. We call the sequence $x_1 y_1 \dots x_k y_k$ a trace of A if $\beta \in h^2(s, \alpha)$. Given two states s of FSM $A = (S, X, Y, h, s_0)$ and t of FSM $B = (T, X, Y, g, t_0)$, state s is said to be equivalent to state t, written $s \leq t$, if $h^2(s, \alpha) = g^2(t, \alpha)$ for each input sequence $\alpha \in X^*$; otherwise, state s is *distinguishable* from state t and the sequence α is said to *distinguish* state s from state t. Machines A and B are *equivalent* if their initial states are equivalent, written $A \equiv B$; otherwise, they are *distinguishable*, written $A \neq B$. An input sequence is said to *distinguish* two machines if it distinguishes their initial states. Machine A is said to be *reduced* if every pair of its states is distinguishable. It is well known that each machine A is equivalent to some reduced observable FSM that is called a reduced observable form of machine A ([Sta72, LPvB94]). In other words, given FSMs A and B over the same input alphabet, A

and B are equivalent if they exhibit the same behavior under all input sequences; otherwise, they are distinguishable. Protocol conformance testing is often formalized as a problem of testing whether an implementation FSM is equivalent to a given reference FSM.

2.2 Composition of FSM's

We consider a system composed by two FSMs, as shown in Figure 1. We will use the expression *system under test* to designate a system composed by some implementation Imp of the component $Comp$ and the context C. For the sake of simplicity, we consider pairwise disjoint sets X, Y, U and V. The system under test has always a single message in transit, i.e. the environment submits the next input only when the system has produced an output to the previous input. We refer to symbols of the alphabet X as external inputs, to symbols of the alphabet Y as external outputs, and to symbols of the alphabets U and V as internal actions. As it is shown in a number of publications ([YCL98]) such a composition is general enough to discuss problems of testing in context. When there are no livelocks in the composition, we can derive the composite FSM of FSMs C and $Comp$ using various algorithms ([PYvBD96, LC97]).

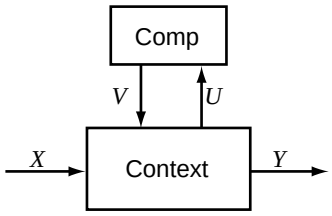

Figure 1: System model

Example. We use as an example of a coffee shop system which is inspired from the one proposed in [PY98]. The coffee shop is composed by a waiter and a coffee machine. We have modified the global behavior of the coffee shop to introduce nondeterminism. The nondeterminism is caused by the context, represented in our example by a forgetful waiter, which may forget a request for coffee, as described below. In order to simplify the description of the components we consider that internal actions are denoted by elements of disjoint sets U and V. The behavior of the coffee shop is as follows: if a customer introduces money (M), the shop answers with thanks (T) and is ready to receive the following inputs: M, which it refuses saying No (N); Espresso Please ($\mathsf{E_p}$), producing two nondeterministic outputs: either Yes (Y), remaining in the same state (forgets the request), or Espresso Served ($\mathsf{E_s}$), returning to the initial state. At the initial state a customer can also request a coffee ($\mathsf{E_p}$), but the answer in this case is sorry (S). The coffee-machine has two inputs and two outputs. When the waiter presses Button (B) at the initial state 0 the coffee-machine responses turning on a Lamp (L). If Button (B) is pressed at the state 1 the coffee-machine produces Espresso (E). An input Coin (C) causes a looping transition and an output Espresso (E) at each state.

The FSMs representing the waiter and coffee-machine are presented in Figure 2.

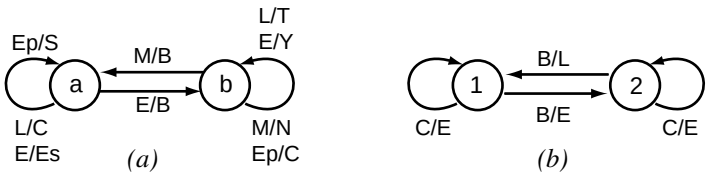

Figure 2: *(a)* Context FSM *Waiter* and *(b)* Component FSM *CoffeeMachine*

The combined system of waiter and coffee machine has no livelocks and its composite FSM RS is shown in Figure 3.

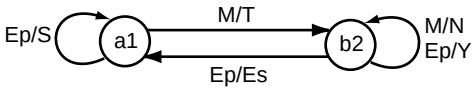

Figure 3: The FSM RS modeling the behavior of the coffee shop

2.3 Fault model for testing in context

We consider a reference system that is a composition of two FSMs, and assume that at most one of the two machines, named the component machine, denoted $Comp$, can be faulty. The other machine, named the context, denoted C, is fault-free. In other words, any implementation system is a composition of the context and some (possibly faulty) implementation component machine Imp combined, as shown in Figure 1. We also assume the behavior of the reference system as well as the behavior of any (possibly faulty) implementation system is described by a reduced observable FSM with at most n states. The set of all possible implementation systems is called a fault domain, and is denoted $\Re_{n,C}$.

To test a system with a nondeterministic behavior each test sequence usually is submitted to the system under test several times, under diverse conditions, until a test engineer verifies the system has produced each possible output to the test sequence. Under this so-called "all-weather conditions" assumption one can use the same notion of a complete test suite for deterministic and nondeterministic machines.

A *test suite* for the reference FSM RS is a finite set of finite input sequences of RS. A test suite for the FSM RS is *complete* w.r.t. the fault domain $\Re_{n,C}$ if for any FSM $B \in \Re_{n,C}$ such that B and the reference machine are distinguishable the test suite has a sequence distinguishing the machines; otherwise, the test suite is not complete w.r.t. the fault domain $\Re_{n,C}$.

One can use a straightforward approach to derive a complete test suite w.r.t. the fault domain $\Re_{n,C}$: explicitly enumerate all FSMs of $\Re_{n,C}$ and for each FSM $B \in \Re_{n,C}$ that is distinguishable from the reference machine determine a sequence distinguishing the machines. The set of all distinguishing sequences is a complete test suite w.r.t. $\Re_{n,C}$. When it is impossible to explicitly enumerate all FSMs of the set $\Re_{n,C}$ we can expand $\Re_{n,C}$ to the fault domain $\Re_n$ comprising any FSM that has a reduced observable form with n states, and use existing methods for complete test derivation w.r.t. the fault domain $\Re_n$. Below we

sketch a method proposed in [LPvB94]; in fact, this method is similar to the well known W or Wp-method for deterministic FSMs when the reference FSM is reduced and completely specified.

Let RS be a reduced connected FSM with n states. A finite subset V, $\mid V \mid \leq n$, including the empty sequence, is called a *state cover* of RS if for each state $s \in S$ there exists an input sequence α such that $s \in h^s(s, \alpha)$. We further denote VX the set obtained by concatenating each sequence of the set V with each sequence of the set X. The set VX is usually called a transition cover of the FSM RS since each transition of the FSM RS is traversed with an appropriate sequence of the set VX. Here we notice that differently from deterministic FSMs the state cover V of a nondeterministic FSM RS can have less sequences than number n of states of RS ([LPvB94]).

Given a state $s \in S$ of the FSM $RS = (S, X, Y, h, s_0)$, a finite set W_s of finite input sequences is said to be a state identifier of s if for each state $p \in S$, $p \neq s$, there exists an input sequence α distinguishing state s from state p, i.e. $h^y(s, \alpha) \neq h^y(p, \alpha)$. An input sequence is said to be a *distinguishing sequence* for the FSM RS if it is a state identifier for every state of RS. Similar to the W-method when a state identifier is fixed for each state of the FSM RS, a procedure for derivation of a complete test suite w.r.t. the fault domain $\Re_n$ comprises two phases.

In the first phase, part T_1 of a test suite is derived. T_1 verifies that an implementation FSM has at least n states and that each state identifier correctly identifies the corresponding state. This part of the test suite is obtained by concatenating each sequence α of the state cover set V with each sequence of the state identifier for each last state of α, that is for each state where α takes the reference FSM, starting from the initial state. If the implementation FSM has the reference output response to each input sequence of the set T_1 then every state of the reference FSM has a corresponding state in the implementation FSM with the same state identifier.

Part T_2 of the test suite derived in the second phase. T_2 checks whether each transition in the implementation FSM is correctly implemented. The set T_2 is obtained by concatenating each sequence $\alpha \in VX$ with each sequence of the state identifier for each state where α takes the reference FSM from the initial state. Merging T_1 and T_2 we obtain a complete test suite $TS = T_1 \cup T_2$ w.r.t. the fault domain $\Re_n$. Each sequence that is a prefix of another sequence can be deleted from TS, without loss of completeness of TS w.r.t. the fault domain $\Re_n$. If we then apply each sequence of TS to an implementation under test, several times, to observe each possible response, and if the set of responses coincides with that of the reference machine then we conclude the implementation has a reference behavior.

Example. The reference machine in Figure 3 is nondeterministic, reduced and observable. We assume the waiter is fault-free and any system with a (possibly faulty) coffee-machine has an observable reduced form with at most two states. An input sequence M is a distinguishing sequence of the reference FSM. We select a state cover set $V = \{\epsilon, \mathsf{M}\}$. Then $T_1 = \{\mathsf{E_p}, \mathsf{M}, \mathsf{ME_p}, \mathsf{MM}\}$ while $T_2 = \{\mathsf{E_pM}, \mathsf{MM}, \mathsf{ME_pM}, \mathsf{MMM}\}$, and $TS = \{\mathsf{E_pM}, \mathsf{ME_pM}, \mathsf{MMM}\}$.

A test suite TS complete w.r.t. the fault domain $\Re_n$ is also complete w.r.t. the fault domain $\Re_{n,C}$. However, it usually is redundant, because some of the test cases included are (unnecessarily) testing only the context, which is assumed fault-free. Furthermore, some

test cases may be testing parts of the implementation already tested by other test cases.

Given a test suite TS, complete w.r.t. the fault domain $\Re_n$, a proper subset T of TS is complete w.r.t. $\Re_{n,C}$ if, for every possible implementation FSM $B \in \Re_{n,C}$, when B has the expected set of output responses to every sequence of T, it has also the expected set of output responses to every sequence of TS. As shown in [YCL98], the detecting power of each test case can be characterized through a set of internal traces that are detectable by the test case. We can delete from TS every test case traversing only transitions of the context, since the set of internal traces that are detectable by this test case is empty, i.e. the test case checks nothing in the component. For example, the suffix MM of the test case MMM tests nothing in the component ([LC97, YCL98]). Therefore, the test suite $\{\mathsf{E_pM}, \mathsf{ME_pM}, \mathsf{M}\}$ is also complete w.r.t. the fault domain $\Re_{n,C}$. If the test case $\alpha \in TS$ traverses transitions in which the component is involved a more detailed analysis of the test case must be performed. In the next sections we show that $TS\backslash\{\alpha\}$ is complete w.r.t. $\Re_{n,C}$ if the following two properties hold: *(i)* there exists a test case $\alpha \in TS\backslash\{\alpha\}$ that induces an unexpected external input for each internal trace that can induce an unexpected external output to α; *(ii)* for each reference output sequence β to α there exists a test case $\alpha \in TS\backslash\{\alpha\}$ which induces an internal trace such that the system under test produces the output response β when α is submitted.

3 Maintaining the fault coverage

In this section, we analyze the conditions to reduce the set TS by deleting some of its sequences without loss of its completeness w.r.t. the fault domain $\Re_{n,C}$. Given an external input sequence α, an internal trace $\gamma = x_1y_1 \dots x_ky_k$ over alphabets U and V is said to be *detectable* by α if, when α is applied to a system under test where the component Imp has the trace $x_1y_1 \dots x_ky_k$, the system produces an unexpected output response to α *YCL98*. Otherwise, a trace is said to be *undetectable* by α. Given a set T of external input sequences, an internal trace γ over alphabets U and V is said to be detectable by T if there exists $\alpha \in T$ such that the internal trace γ is detectable by α.

In [YCL98] we show that any prolongation of a trace detectable by α is also detectable by α, and that the set of all internal traces detectable by α, in the case of systems composed by deterministic context and component FSMs, is a regular set ([HU79]). The same results are valid in the case of nondeterministic context and/or component.

3.1 Determining detectable internal traces

Procedure 1 below describes a step by step derivation of the regular set of detectable internal traces for given context. The key construction of the procedure is the acceptor $LC(\alpha)$ of all possible traces that can be induced by an external input sequence when the context is combined with any implementation of the component machine Comp. Traces detectable by α are designated in the $(U \cup V)$-projection of the acceptor $LC(\alpha)$ by a dead state *fail*. Below we briefly sketch how the acceptor $LC(\alpha)$ can be constructed.

Given an external input sequence $\alpha = x_1 \dots x_k$, the acceptor $LC(\alpha)$ can be derived step by step by use of the deterministic Label Transition System (LTS) LC representing

all traces of the context C. States of the acceptor $LC(x_1 \dots x_k)$ are states of LC. Given an input x_1, we construct the acceptor $LC(x_1)$ starting from the initial state of the LC. There is a transition labeled with x_1 from the initial state to state p if x_1 takes LC from the initial state to state p. For two intermediate states p and r, there is a transition labeled with internal action $a \in U \cup V$ if there is a transition labeled with a from state p to state r in LC. There is a transition labeled with $y \in Y$ in the $LC(x_1)$ from intermediate state p to a final state if there exists an outgoing transition labeled y from state p. The acceptor $LC(x_2)$ is constructed at each final state of the acceptor $LC(x_1)$ and so on. The final states of the acceptor $LC(x_k)$ are declared the final states of the acceptor $LC(x_\alpha)$.

Procedure 1 *Derivation of a regular set of detectable internal traces.*

Input: The composite FSM RS of a reference system, the deterministic LTS LC representing all traces of the context C and an external input sequence α.

Output: The regular set $D(\alpha)$ of internal traces detectable by α.

Step 1. Construct the acceptor $LC(\alpha)$ using the deterministic context LTS LC.

Step 2. For each path of the acceptor $LC(\alpha)$ from the initial state to a final state such that the projection of the sequence labeled the path is not a trace of the reference FSM, replace the final state with a final state *fail*.

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

Step 4. Construct the $(U \cup V)$-projection of the obtained acceptor by a subset construction, replacing with a designated fail-state without outgoing transitions each subset that has the fail-state. Construct the regular set $D(\alpha)$ as the set of all sequences of the $(U \cup V)$-projection labeled paths from the initial to the *fail* state.

By construction of the set $D(\alpha)$, the following statements hold.

Proposition 1 *Given a reference system RS and an implementation system IS with a component Imp, the system IS produces an unexpected response to the input sequence α if and only if the set of traces of the machine Imp intersects the set $D(\alpha)$ derived by Procedure 1 ([YCL98]).*

Corollary 2 *Given a reference FSM RS, an implementation system IS, a complete test suite TS w.r.t. the fault domain $\Re_{n,C}$ and a sequence $\alpha \in TS$, let the system IS have an unexpected response to the sequence α. If for each trace $u_1v_1 \dots u_kv_k \in D(\alpha)$ there exists a sequence $\gamma \in TS \backslash \{\alpha\}$ such that the set $D(\alpha)$ comprises a prefix of the trace $u_1v_1 \dots u_kv_k$, then the system IS has an unexpected response to some sequence of the set $TS \backslash \{\alpha\}$.*

In fact, let all the conditions of Corollary 2 hold. If the IS has an unexpected response to the sequence α then the set of traces of the component Imp has a trace $u_1v_1 \dots u_kv_k$ of

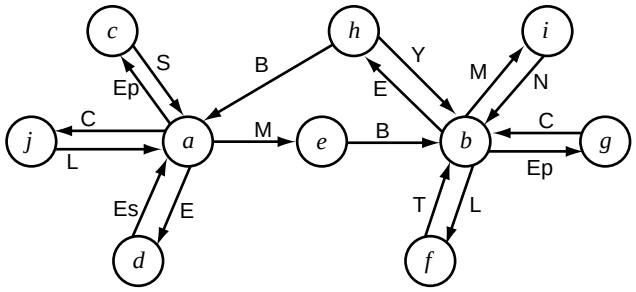

Figure 4: The context LTS LC

the set $D(\alpha)$ (Proposition 1), along with its prefix $u_1v_1 \dots u_lv_l, l \leq k$, being in the set $D(\gamma)$ for some sequence $\gamma \in TS\backslash\{\alpha\}$ i.e. IS has an unexpected response to the sequence γ.

Example. Consider an input sequence $\mathsf{M} \in TS$. The deterministic context LTS LC is shown in Figure 4. The acceptor $LC(\mathsf{M})$ and its $(U \cup V)$-projection with the designated fail-state are shown in Figures 5 and 6. If the component implementation Imp has a trace BE then the implementation system IS has an unexpected output response Y to input M. On the other hand, if the component implementation Imp has no trace BE then the implementation system IS always produces the expected output response T to the input M. Thus, the implementation system IS always produces the expected output response T to the input M if and only if the component implementation Imp answers with output E to the input B. Thus, $D(\mathsf{M}) = \{\mathsf{BE}\}$. Similarly, we obtain $D(\mathsf{E_pM}) = \{\mathsf{BE}\}$ and $D(\mathsf{ME_pM}) = \{\mathsf{BE}, \mathsf{BLCL}, \mathsf{BLCEB(LC)^*EBE}\}$. By direct inspection, one can assure that if the implementation system IS has only expected output responses to the input sequence $\mathsf{ME_pM}$ then the implementation component machine Imp has no traces of the set $D(\mathsf{ME_pM})$, i.e. IS has the expected output responses to the input sequences $\mathsf{E_pM}$ and M.

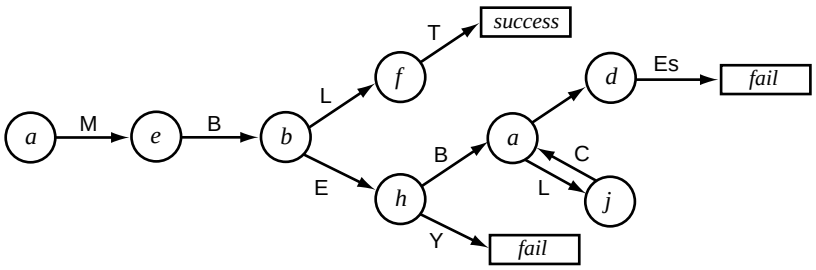

Figure 5: Acceptor LC(M)

Corollary 2 specifies a necessary and sufficient condition for minimizing a test suite while maintaining its fault detection power when the context and the component are deterministic. Although this condition is insufficient in the case of nondeterministic systems, since the minimized test must also accept all possible behaviors accepted by the original test suite. That property is analyzed in the next section.

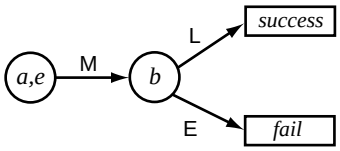

Figure 6: $(U \cup V)$–projection of acceptor LC(M)

3.2 Preserving all possible behaviors

Given an external input sequence $\alpha = x_1 \dots x_k$, let $LC(\alpha)$ be an acceptor of all internal traces that can be induced in the context when α is submitted to an implementation system. Let also $\beta = y_1 \dots y_k$ be an output sequence that can be produced by the reference system in response to α. We denote $C(\alpha, \beta)$ the regular set of $(U \cup V)$-projections of all sequences that label paths from the initial to the final state in $LC(\alpha)$ and have the $(X \cup Y)$-projection $x_1 y_1 \dots x_k y_k$. If a system under test has produced an output response β to the sequence α, it means that the set of traces of the implementation component machine intersects the set $C(\alpha, \beta)$.

Procedure 2 *Derivation of the regular set of all internal traces which are possible in an implementation system with a given reference trace.*

Input: The composite FSM RS of a reference system, the LTS LC representing all traces of the context C, an external input sequence α and the output response β of the reference system RS to α.

Output: The regular set $C(alpha, \beta)$ of all internal traces which are possible to occur in an implementation system with the reference trace (α, β).

Step 1. Construct the acceptor $LC(\alpha)$ using the deterministic context LTS LC.

Step 2. For each path of the acceptor $LC(\alpha)$ from the initial state to a final state such that the projection of the sequence labeled the path is a trace (α, β), replace the final state with a designated state *success*.

Step 3. Construct the $(U \cup V)$-projection of the obtained acceptor by a subset construction replacing with a designated state *success* each subset that has the state *success*. Derive the regular set $C(\alpha, \beta)$ as the set of all sequences of the $(U \cup V)$-projection labeled paths from the initial to the state *success*.

Proposition 3 *Given two external input sequences α and β, the sets of output responses $h^y(s_0, \alpha)$ and $h^y(s_0, \delta)$ of the reference system RS to the sequences α and δ, let IS be an implementation system such that the set of output responses of IS to δ coincides with $h^y(s_0, \delta)$. If for every $\beta \in h^y(s_0, \alpha)$ there exists a sequence $h^y(s_0, \delta)$ such that each sequence in the set $C(\delta, \lambda)$ has a prefix in the set $C(\alpha, \beta)$ then the set of output responses of IS to α coincides with $h^y(s_0, \alpha)$.*

As a corollary to Proposition 3, another condition for a proper subset T of the complete test suite TS to provide a complete set of reference outputs to each test case in TS can be established.

Corollary 4 *Given the reference FSM RS, an implementation system IS, a complete test suite TS w.r.t. the fault domain $\Re_{n,C}$ and a sequence $\alpha \in TS$, let the system IS have only expected output responses to the sequence α. Let also for each reference output response β to the sequence α, there exist a sequence $\delta \in TS\backslash\{\alpha\}$ and $\lambda \in h^y(s_0, \delta)$ such that each trace in the set $C(\delta, \gamma)$ has a prefix in the set $C(\alpha, \beta)$. If for each sequence $\delta \in TS\backslash\{\alpha\}$, the set of output responses of IS to δ coincides with that of the reference system then the set of output responses of the system IS to the sequence α also coincides with that of the reference system.*

Example. The set $C(\mathsf{ME_pM}, \mathsf{TYN}) = \{\mathsf{BLCE}\}$ while $C(\mathsf{ME_pM}, \mathsf{TE_sT}) = \{\mathsf{BLCEB(LC)^*EBL}\}$. In other words, an implementation system IS has all expected output responses to the input sequence $\mathsf{ME_pM}$ if and only if the implementation component machine has the trace BLCE as well as at least one trace of the regular set $\mathsf{BLCEB(LC)^*EBL}$.

4 Minimization of a test suite

By combining the results of Corollaries 2 and 4 we obtain a sufficient condition for a proper subset of a complete test suite to be also complete w.r.t. the fault domain $\Re_{n,C}$.

Proposition 5 *Given a reference FSM RS, a complete test suite TS w.r.t. the fault domain $\Re_{n,C}$, let $\alpha \in TS$. The set $TS\backslash\{\alpha\}$ is also complete w.r.t. $\Re_{n,C}$ if the following conditions hold.*

a) For every trace $u_1v_1 \dots u_kv_k \in D(\alpha)$, there exists $\delta \in TS\backslash\{\alpha\}$ such that the set $D(\delta)$ comprises a prefix of the trace $u_1v_1 \dots u_kv_k$.

b) For every reference output response $\beta \in h^y(s_0, \alpha)$ there exist $\delta \in TS\backslash\{\alpha\}$ and $\lambda \in h^y(s_0, \delta)$ such that for every trace $u_1v_1 \dots u_kv_k \in C(\delta, \lambda)$, the set $C(\alpha, \beta)$ comprises a prefix of the trace $u_1v_1 \dots u_kv_k$.

Example. In our working example, only a single input sequence $\mathsf{ME_pM}$ of the test suite induces two reference output responses. Thus, the set $\{\mathsf{ME_pM}\}$ is a test suite complete w.r.t. the fault domain $\Re_{n,C}$.

Proposition 5 shows that a given test suite can be minimized by examining the regular sets detectable by the test cases and the sets of internal traces providing reference output sequences. The problem of comparing arbitrary regular expressions is out of the scope of this paper. When the regular sets C and D derived by Procedures 1 and 2 are finite; in that case, similarly to [YCL98], we can reduce the problem of test minimization to the problem of determining a minimal column coverage in a Boolean matrix, as described below.

Given a complete test suite TS, let $D(TS)$ be the set of internal traces detectable by the set TS derived by Procedure 1. For each $\alpha \in TS$ that induces at least two reference output

responses and each $\beta \in h^y(s_0, \alpha)$, we derive the set $C(\alpha, \beta)$ of internal traces which provide the external trace (α, β) (Procedure 2) and denote $C(TS)$ the collection of such sets. We construct a Boolean matrix B as follows. Rows of the matrix B correspond to sequences of the prefixes of all sequences in the test suite TS; columns of B correspond to the items in $D(TS)$ and in $C(TS)$. That is, the columns of B are all internal traces detectable by TS, and all subsets $C(\alpha, \beta)$ of internal traces induced by each trace (α, β) of the reference FSM RS for each $\alpha \in TS$ providing at least two reference output sequences.

Element b corresponding to a test sequence $\alpha \in TS$ and an internal trace $\gamma \in D(TS)$ has value '1' if and only if there exists a prefix of γ detectable by α. Element b corresponding to a test sequence $\alpha \in TS$ and a subset $C(\delta, \lambda), \delta \in TS$, has value '1' if and only if there exists a response β of the reference FSM RS to α such that each sequence in the set $C(\alpha, \beta)$ has a prefix in the set $C(\delta, \lambda)$. The set of rows of the matrix corresponding to its minimal column coverage is a test suite complete w.r.t. the fault domain $\Re_{n,C}$.

5 Conclusion

In this paper, the approach proposed in [YCL98] is extended to a system of communicating nondeterministic FSMs. The system under test is composed of two FSMs, a context that is assumed to be fault-free and a component that needs testing. We assume that the behavior of both the reference and the implementation system are described by observable FSMs. In order to minimize a given test suite, complete under the above assumptions, we construct two regular sets. One of them is the set of all internal traces detectable by each test case, i.e. a set of internal traces which cause an unexpected output response of an implementation system. Another regular set is the set of all internal traces that cause a given reference output to the test case. If a system under test produces every reference output response to a test case one guarantees that the component machine has at least one trace of such set for each reference output response. After the regular sets have been derived the problem of determining a minimal subset of a given test suite that is also complete w.r.t. the chosen fault domain can be solved as a problem of finding a minimal column coverage of a Boolean matrix. We can apply the approach for test minimization w.r.t. the reduction relation. In this case, similar to a system of communicating deterministic finite state machines the Procedure 2 becomes unnecessary. The proposed method can also be used to reduce a test suite given by a human expert while preserving its fault detection power under an assumption that the context is fault-free.

References

[Cho78] T. S. Chow. Test software design modeled by finite state machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, 1978.

[FvBK+91] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghendamsi. Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, 1991.

[HU79] J. E. Hopkroft and J. D. Ulman. *Introduction to automata theory, languages and computation*. Addison-Welsey, NY, 1979.

[ISO91] ISO. *Information technology, Open systems interaction, Conformance testing methodology and framework*. International Standard IS-9646, 1991.

[LC97] L. P. Lima and A. R. Cavalli. A pragmatic approach to generating test sequences for embedded systems. In *10th IWTCS*, pages 125–140, 1997.

[LC98] L. P. Lima and A. R. Cavalli. Application of embedded testing methods to service validation. In *submitted to 2nd IEEE Intern. Conf. On Formal Engineering methods*, 1998.

[LPvB94] G. Luo, A. Petrenko, , and G. v. Bochmann. Selecting test sequences for partially-specified nondeterministic finite state machines. In *7th IWTCS*, pages 95–110, 1994.

[LSKP96] D. Lee, K. K. Sabnani, D. M. Kristol, and S. Paul. Conformance testing of protocols specified as communicating finite state machines – a guided random walk based approach. *IEEE Transations on Communications*, 44(5):631–640, 1996.

[LY96] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines, a survey. *IEEE Transactions*, 84(8):1090–1123, 1996.

[PY97] A. Petrenko, , and N. Yevtushenko. Testing faults in embedded components. In *10th IWTCS*, pages 125–140, 1997.

[PY98] A. Petrenko, , and N. Yevtushenko. Solving asynchronous equations. In *Joint International Conference FORTE/PSTV98*, pages 231–247, 1998.

[PYvB96] A. Petrenko, N. Yevtushenko, , and G. v. Bochmann. Fault models for testing in context. In *1st Joint International Conference FORTE/PSTV96*, pages 125–140, 1996.

[PYvBD96] A. Petrenko, N. Yevtushenko, G. v. Bochmann, and R. Dssouli. Testing in context: framework and test derivation. *Computer communications*, 19:1236–1249, 1996.

[Sta72] P. H. Starke. *Abstract automata*. American Elsevier Publishing Company, Inc. New York, 1972.

[Vas73] M. P. Vasilevsky. Failure diagnosis of automata. *Cybernetics*, (4):653–665, 1973.

[VCI89] S. T. Vuong, W. W. L. Chan, and M. R. Ito. The uio-method for protocol test sequence generation. In *IFIP TC6 Second International Workshop on Protocol Test Systems*, pages 161–175, 1989.

[YCL98] N. Yevtushenko, A. R. Cavalli, and L. P. Lima. Test minimization for testing in context. In *11th IWTCS*, pages 127–145, 1998.

[YL95] M. Yannakakis and D. Lee. Testing finite state machines: fault detection. *Journal of Computer and System Sciences*, (50):209–227, 1995.