Fault detection in embedded components

A. Petrenko and N. Yevtushenko †

CRIM, Centre de Recherche Informatique de Montréal,
1801 Av. McGill College, Montréal, H3A 2N4, Canada,
Phone: (514) 840-1234, Fax: (514) 840-1244, petrenko@crim.ca
† *Tomsk State University,*
36 Lenin str., Tomsk, 634050, Russia,
yevtushenko.rff@elefot.tsu.tomsk.su

Abstract

We address in this paper the problem of detecting faults located in a given component embedded within a composite system. The system is represented as two communicating FSMs, a component FSM inaccessible for testing and a context machine that models the remaining part of the system which is assumed to be correctly implemented. We elaborate a systematic approach for deriving external tests which can detect all predefined types of faults in the embedded component. The approach is based on the construction a proper characterization of the conforming behavior of the component in context, derivation of internal tests and translation into external tests.

Keywords

Communicating FSMs, fault models, conformance testing, embedded testing, test derivation

1 INTRODUCTION

The model of communicating state machines, see e.g., [Boch78], [BrZa83], is widely used for development of complex systems. It serves as an underlying model for description techniques such as Statecharts, ROOM, ESTEREL, SDL. One of the important issue is test derivation from a formal specification in the form of communicating state machines. A straightforward solution is to construct a global composed machine from a reachability graph such that describes the behavior of a system at points accessible for testing and apply existing test derivation methods developed for FSMs. The behavior of a system even consisting of deterministic components may be nondeterministic and a test derivation method which can treat nondeterministic I/O FSMs should be used [LBP94]. This approach suffers from several drawbacks. First, even if each component of the system is given as an I/O FSM, the global I/O machine may not exist due for example, livelocks. A number of verification methods and tools could be used to check properties of the given system, so it is reasonable to assume that tests should be derived from a verified system of communicating state machines such that its composed machine exists. Second, the number of states in the composed machine (assuming that we are able to construct it) may easily trigger tests with a high fault coverage to explode. Two main approaches have been tried to alleviate the test explosion effect.

According to the first approach, systematic test derivation with fault coverage is avoided, transition coverage of individual component machine is attempted instead. This could be achieved a partial exploration of the composed machine either by adopting a random walk [West86], see [LSKP96], by generating a certain part of the entire composed machine comprising transitions chosen for testing [HLS96] or a reduced composed machine [KoTa95]. The advantage of this approach is that the need for global machine construction is obviated. However, the fault detection ability of the approach is unknown.

The second approach is driven by a divide-and-conquer strategy and is closely related to the problem of submodule construction, known also as redesign, plant-controller, or equation solving, where we are required to construct the specification of a submodule X when specifications of the overall system and of all submodules except X are given [MeBo83], [QiLe91], [ABBD95], [LJK95], [HeBr95]. A given system of communicating FSMs is viewed in two parts, one part (an embedded component) is to be tested and the other (context of the component) is assumed to be error-free. The main issue here is how to systematically derive tests tuned for the embedded component (testing in context). The basic idea is to reduce testing in context to testing in isolation so that existing methods could become fully applicable. Once this problem is solved we may similarly proceed deriving tests for the remaining part of the system (the target component and context switch their roles). Since faults usually do not affect all the components of a system the resulting test suite would normally have a high fault coverage, while test explosion effect is alleviated. This approach has been elaborated in [PYLD93], [PYD94], [PYBD96] and [PYB96]. Here we continue that work for providing systematic methods for test derivation from communicating state machines.

The rest of the paper is organized as follows. In Section 2, we briefly summarize the results of [PYBD96] and [PYB96] related to this work. The novel parts are presented in Section 3 and 4. Section 3 gives a method for constructing a so called embedded equivalent of the component in context which explicitly characterizes all implementations conforming to a given specification in context and facilitates test derivation. Section 4 discusses the problem of translating internal tests derived from the embedded equivalent into external tests. Two approaches to solve the problem are proposed. We conclude in Section 5 with a discussion of future work.

2 FRAMEWORK FOR TESTING IN CONTEXT

2.1 Finite state machines

A finite state machine (FSM) is a completely specified initialized (possibly nondeterministic) Mealy machine which can be formally defined as follows. A *finite state machine* A is a 5-tuple (S, X, Y, h, s_0), where S is a 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; and h - a behavior function $h: S\times X \rightarrow \wp(S\times Y)\backslash\varnothing$, where $\wp(S\times Y)$ is the powerset of $S\times Y$ [Star72]. The machine A becomes deterministic when $|h(s, x)|=1$ for all $(s, x)\in S\times X$.

We extend the behavior function to a function on the set X^* of all input sequences containing the empty sequence ε, i.e., $h: S\times X^* \rightarrow \wp(S\times Y^*)\backslash\varnothing$. Assume $h(s, \varepsilon) = \{(s, \varepsilon)\}$ for all $s\in S$, and suppose that $h(s, \beta)$ is already specified. Then $h(s, \beta x) = \{ (s', \gamma y) \mid \exists s''\in S\ [(s'', \gamma) \in h(s, \beta) \wedge (s', y) \in h(s'', x)] \}$. Given a sequence α over the alphabet $X\cup Y$, we use α^X to denote the X-projection of α that is obtained by deleting all symbols $y\in Y$ from the sequence α.

The function h^1 is the next state function, while h^2 is the output function of A, where h^1 is the first and h^2 is the second projection of h, i.e., $h^1(s, \alpha) = \{ s' \mid \exists\ \beta \in Y^*\ [(s', \beta) \in h(s, \alpha)] \}$, $h^2(s, \alpha) = \{\beta \mid \exists\ s' \in S\ [(s', \beta) \in h(s, \alpha)] \}$ for all $\alpha\in X^*$. We use $h^1_\beta(s, \alpha)$ to denote the set of states reached by the machine when it executes I/O sequence α/β starting from state s. Given two states s of the FSM A and r of the FSM $B= (T, X, Y, H, t_0)$, and a set $V\subseteq X^*$; state r is said to be a *V-reduction* of s, written $r \leq_V s$, if for all input sequences $\alpha\in V$ the condition $H^2(r, \alpha) \subseteq h^2(s, \alpha)$ holds; r is not a V-reduction of s, $r \not\leq_V s$, if there exists an input sequence $\alpha \in V$ such that $H^2(r, \alpha) \not\subseteq h^2(s, \alpha)$. States s and r are *V-equivalent* states, written $s \cong_V r$, iff $s \leq_V r$ and $r \leq_V s$. On the class of deterministic machines, the above relations coincide. We denote $\leq$ the V-reduction in the case where $V=X^*$, similarly, $\cong$ denotes the equivalence relation. Given two machines, A and B, B is a reduction of A, written $B\leq A$, if the initial state of B is a reduction of the initial state of A. If $B\leq A$ and B is deterministic then it is referred to as a D-reduction of A. Similarly, the equivalence relation between machines is defined. $B\cong A$, iff $B\leq A$ and $A\leq B$. The equivalence and reduction relations serve as conformance relations between implementations and their FSM specifications for deriving test suites with guaranteed fault coverage [SiLe89], [PBD93], [YaLe95], [PYB96a].

A *fault model* is a triple $<A, \sim, \mathfrak{I}>$ [PYB96], where A is a reference specification, a set $\mathfrak{I}$ is the fault domain that is a set of possible implementations defined over the same input alphabet as the specification, and $\sim$ is a conformance relation. In this paper, we consider $\sim \in \{\cong, \leq\}$. A *complete* test suite w.r.t. the fault model is a finite set E of finite input sequences such that for all $B \in \mathfrak{I}$, $B \not\sim A$ implies $B \not\sim_E A$.

If the fault domain is an arbitrary finite set $\mathfrak{I}$ of implementation machines then in order to derive a complete test suite w.r.t. the fault model a traditional method (mutant killing technique) could be used. For each FSM $B \in \mathfrak{I}$, we derive an input sequence that distinguishes B from the reference specification A whenever they are not equivalent (or B is not a reduction of A). The union of input sequences over all machines $B \in \mathfrak{I}$ gives a desired test suite. Because of its complexity, such a solution is feasible for a small number of faults to be detected, for example for single output faults. At the same time, there are certain fault models for which there is no need to explicitly enumerate machines of the fault domain. For these fault models, a complete test suite is derived based on the properties of the specification machine A. As an example, we could mention a classical (black-box) fault model $< A, \cong, \mathfrak{I}_m(X)>$ where A is a completely specified and deterministic FSM, and $\mathfrak{I}_m(X)$ is the set of all FSMs over the input alphabet X of A with at most m states. A number of competing methods exist, see e.g., [SiLe89], [PBD93], [YaLe95]. As is shown in [PYB96], a similar approach can be taken to devise fault models and to derive complete tests for embedded components. In this paper, we propose new methods for testing in context such that allow to obviate an expensive mutant killing technique.

2.2 Model of a system with the embedded component

Many compound systems are typically specified as a collection of communicating FSMs. As noticed in [PYBD96] the system of two communicating FSMs (IUT and context), connected as shown in the upper part of Figure 1, is general enough to discuss problems related to testing an embedded component.

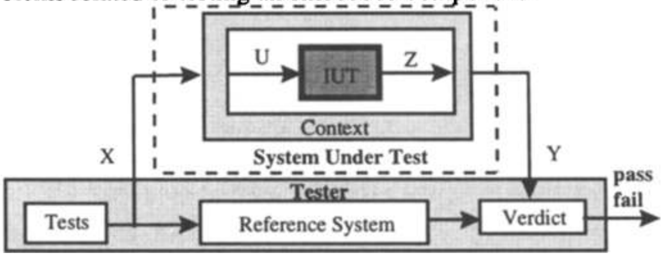

Figure 1 Architecture for testing the embedded component (IUT).

We assume that we are given an FSM *Spec* which represents the behavior of the component (IUT) embedded within the system that should be tested, while a machine C, called the *context* machine, is a composed machine of all components of the system, except the component of interest, that are assumed fault-free. As in [PYB96], we assume that the sets X, U, Z, and Y of actions are pairwise disjoint. Two (deterministic) FSMs are communicating asynchronously via bounded input queues where actions are stored. We assume that the system at hand has a single message in transit, i.e. a next external input x is submitted to the system only after it has produced an external output y to the previous input. Under these assumptions, the collective behavior of two communicating FSMs can be described by means of a *product* machine and a *composed* machine. The product machine *Spec*×*C* is represented by a graph of global states, obtained by performing reachability computation [BrZa83]. It is in fact, a labeled transition system which represents the joint behavior of all components. If the product machine *Spec*×*C* has a cycle labeled only with internal actions from the alphabet $U \cup Z$ then the system falls into livelock when an appropriate input sequence is applied, i.e. the system cannot produce an external output. In this case, the system's behavior cannot be described by an I/O FSM and we conclude that the composed machine does not exist. Otherwise, a composed machine $RS = Spec \circ C$ can be obtained by hiding of all internal actions in the product machine, determinizing the obtained LTS and by pairing inputs with subsequent outputs [PYB96], [PYBD96].

Example. Consider the system [PYB96] of context and component machines, shown in Figure 2. The composed machine $RS = Spec \circ C$ is shown in Figure 2(c).

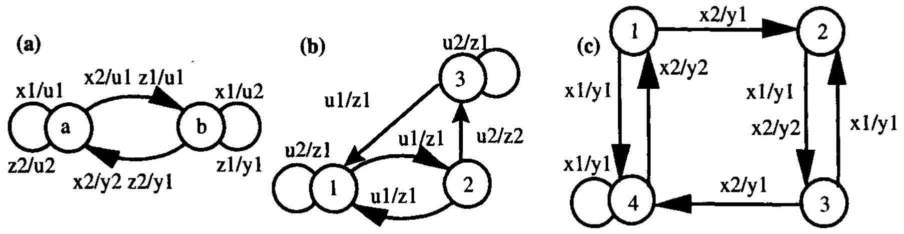

Figure 2 The context *C* (a), component *Spec* (b), and the composed machine *RS* (c).

2.3 Explicit fault model for testing in context

Testing in context is based on the test architecture shown in Figure 1. We assume that the tester executes test cases simultaneously against the system under test and its specification, called the reference system. The reference system is modeled by the composed machine $RS = Spec \circ C$. The embedded component (IUT) is the target of tests. The context does not need to be tested. Verdicts are produced by a part of the tester called a verdict machine. The verdict machine produces the verdict **fail** and enters a state FAIL when output actions of a system under test and reference system do not coincide or the system under test falls into livelock. No communication between the component and context can be observed or controlled.

Based on the test architecture (Figure 1), we define a fault model for deriving complete test suites as in [PYB96]. Let $\mathfrak{I}_m(U, Z)$ denote the set of all implementation FSMs *Imp* over alphabets *U* and *Z* with at most *m* states such that $Imp \circ C$ exists. Then the triple $<RS, \cong, \mathfrak{I}_m(U, Z) \circ C>$ where $\mathfrak{I}_m(U, Z) \circ C = \{Imp \mid Imp \in \mathfrak{I}_m(U, Z)\}$, is called the *explicit fault model* for testing in context. In this paper, we attempt to elaborate a systematic method for deriving a test suite complete w.r.t. the explicit fault model. In [PYB96], we have considered a number of fault models relevant for testing in context, however this problem was left open.

2.4 Approximation of the component's behavior

To systematically derive tests for the embedded component we need a complete and concise characterization of detectable and undetectable faults. This is what we call the approximation of component's behavior in the given context C [PYBD96], [PYB96] that completely describes the permissible behavior of the embedded component w.r.t. any external input sequence. Below we briefly summarize its construction.

A trace of the embedded component is permissible if it is a valid trace of its specification *Spec*. If it is not in *Spec* then, depending on a particular external input sequence, it may be permissible or forbidden. The verdict machine producing the *fail*-verdict in response to the external input sequence indicates that the behavior of the component is forbidden. We formalize the notions of permissible and forbidden traces of the embedded component as follows.

A trace $\beta/\gamma \in U^*/Z^*$ is ***forbidden*** w.r.t. an external input sequence $\alpha \in X^*$ if there exists a prefix $\beta_1 \ldots \beta_k/\gamma_1 \ldots \gamma_k$ of β/γ such that for an appropriate prefix $\alpha_1 \ldots \alpha_k$ of the sequence α it holds that the U-projection of the output sequence of the context C to $\alpha_1\gamma_1 \ldots \alpha_k\gamma_k$ is equal to $\beta_1 \ldots \beta_k$ while its Y-projection is not equal to the output sequence of the reference system RS to $\alpha_1 \ldots \alpha_k$. Trace β/γ is said to be ***permissible*** w.r.t. the external input sequence α otherwise. Trace β/γ is permissible if it is permissible w.r.t. all external input sequences. In other words, the trace β/γ is forbidden w.r.t. an external input sequence α if every system composed of a component that contains trace β/γ is not equivalent to the reference system RS w.r.t. α, i.e. α can be considered as an external test detecting any nonconforming implementation of *Spec* with trace β/γ.

The idea of constructing the approximation is based on the test architecture presented in Figure 1. To capture all possible behavior of the embedded component we replace it with a chaos machine *Ch* over the alphabets U and Z that has just one state [PYB96]. The chaos machine is nondeterministic, it produces all possible outputs z in response to each input u. We construct the product machine $Ch \times C \times RS \times Ver$ as an LTS, hide all actions Y and verdicts in the obtained LTS and determinize it. The resulting LTS is transformed back to an FSM, denoted $[[Spec]]_C$, in alphabets $X \cup U$ and $Z \cup \{null, fail\}$. Any global state where the verdict machine is in a fail-state is a designated state FAIL of $[[Spec]]_C$. An external input x is coupled with the output *fail* and labels a transition to the state FAIL if all subsequent internal actions lead to the state FAIL; otherwise it is coupled with the output *null.* The remaining internal inputs u are paired with the internal outputs z. "Don't care" transitions of the obtained FSM are specified as transitions to another designated state TRAP. Specifically, if an external input x causes a "don't care" transition from a particular state then the machine has a transition to the state TRAP labeled $x/null$, for an input u a corresponding transition to the TRAP state is labeled with input u and each internal output $z \in Z$. Intuitively, the TRAP state indicates that any behavior of the component machine when the FSM $[[Spec]]_C$ trapped to this state, is permissible since it cannot be executed. Any behavior leading to the FAIL state is forbidden, since it results in a wrong external output. For more details on the construction of the approximation of the component in context the reader is referred to [PYBD96]. Figure 3 shows the approximation $[[Spec]]_C$ for our example. The

FSM $[[Spec]]_C$ captures the most essential for testing aspect of the behavior of the whole system shown in Figure 1. In particular, the verdict machine in response to a particular external input sequence produces the *fail*-verdict in a current global state of the system if and only if the FSM $[[Spec]]_C$ reaches the state FAIL. This property of the approximation is formally stated as follows.

Proposition 2.1. Given the approximation $[[Spec]]_C = (S, X \cup U, Z \cup \{fail, null\}, h, s_0)$ and trace $\beta/\gamma \in (U/Z)^*$, the trace β/γ is forbidden iff there exists an I/O sequence α/δ of $[[Spec]]_C$ with the $(U \cup Z)$-projection β/γ that takes $[[Spec]]_C$ from the initial state to the state FAIL.

Given a forbidden trace β/γ, we denote $\alpha(\beta/\gamma)$ the input part of an I/O sequence of $[[Spec]]_C$ that has the $(U \cup Z)$-projection β/γ and takes $[[Spec]]_C$ from the initial state to the state FAIL. The trace β/γ is forbidden w.r.t. the X-projection of α. The approximation of the component in context characterizes the relationship between deviations in the behavior of the embedded component and external input sequences capable of revealing a fault through the context. However, its shortcoming is that existing test derivation methods cannot be directly applied to derive external tests. At the same time, as we are going to demonstrate in the subsequent section, it can be further transformed into another machine allowing for a direct use of these methods.

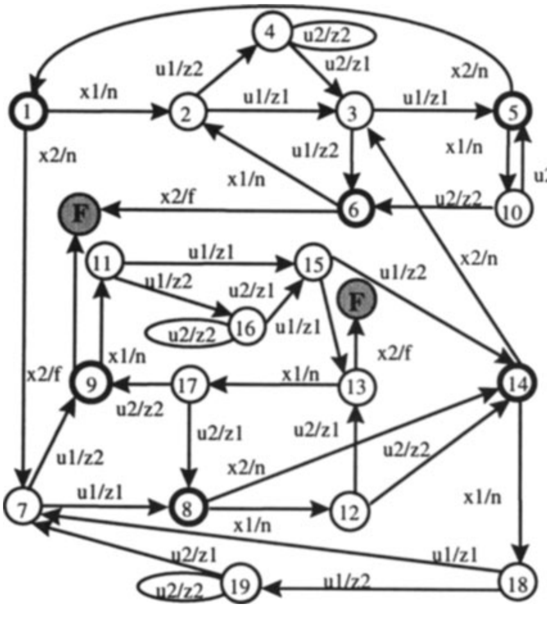

Figure 3 The approximation of *Spec* in context. State TRAP as well as its incoming transitions are not shown, state **F** is the FAIL state.

3 EMBEDDED EQUIVALENT OF A COMPONENT MACHINE

In order to use regular methods for test derivation we now would like to transform the approximation $[[Spec]]_C$ into an FSM such that all its I/O sequences in alphabets U and Z of *Spec* are permissible w.r.t. every possible external input sequence.

Equivalently, we define a machine by excluding from the set $(U/Z)^*$ all traces β/γ such that are forbidden w.r.t. some external input sequence in X^*. Let Tr be the set of traces of a machine and $[[Spec]]_C = (S, X\cup U, Z\cup\{fail, null\}, h, s_0)$.

An FSM is said to be the *embedded equivalent* of the component *Spec* in context *C*, denoted $EE = (P, U, Z\cup\{fail\}, H, p_0)$ if its traces in $Tr(EE)$ over the inputs U and outputs Z satisfy the conditions of Proposition 2.1, namely:

$$\forall \beta/\gamma \in (U/Z)^* \ (\beta/\gamma \text{ is forbidden}) \Leftrightarrow \beta/\gamma \notin Tr(EE) \vee H^1_\gamma(p_0,\beta) = \{\text{FAIL}\}.$$

The idea of transforming the approximation $[[Spec]]_C$ into the embedded equivalent is to hide all external inputs X and to group its states into subsets such that all external inputs cause transitions in the FSM $[[Spec]]_C$ within the same subset, making sure that all forbidden traces are removed. The situation is somewhat similar to a classical problem of determinizing a nondeterministic finite automaton (the subset construction) [HoUl79], where all non-observable actions have to be removed while preserving all the traces of a given automaton. In fact, as in our case, all states reached from a given state through internal transitions (corresponding to external inputs) could be merged to form a single state of resulting machine. The essential difference is that in our case, we should retain only traces that are permissible w.r.t. all external input sequences, i.e. that are common for all states reached from the same state after non-observable actions. In other words, we should determine the intersection of such traces for each state instead of collapsing traces. As the intersection may sometimes become empty we use a designated output *fail* and state FAIL in the embedded equivalent to indicate that a certain common trace can no longer be extended, since there exists an external input sequence that "forbids" any extension. To formalize the procedure we need the following definition.

Given the FSM $[[Spec]]_C = (S, X\cup U, Z\cup\{fail, null\}, h, s_0)$, a set B of states of $[[Spec]]_C$ is said to be *closed* (w.r.t. external inputs) if $h^1(s, x) \subseteq B$ holds for every $s \in B$ and $x \in X$. For a subset $B \subseteq S$, a minimal by inclusion closed set including B is called the *closure* of B.

We present the procedure using our example (Figure 3). The closure of the initial state of $[[Spec]]_C$ is the set $\{1, 2, 7\}$ which is the initial state of an FSM *EE*. In $[[Spec]]_C$, inputs u_1 and u_2 cause transitions to the TRAP state from state 1. State 2 has the following transitions: $2\text{-}u_1/z_1\text{->}3$, $2\text{-}u_1/z_2\text{->}4$. State 7: $7\text{-}u_1/z_1\text{->}8$, $7\text{-}u_1/z_2\text{->}9$. Both states have transitions caused by input u_2 to state TRAP.

Consider input u_1. We have $\bigcap_{s\in\{1,2,7\}} h^2(s, u_1) = \{z_1, z_2\}$.

For the output z_1, we find the union of states $\bigcup_{s\in\{1,2,7\}} h^1_{z_1}(s, u_1) = \{3, 8, \text{TRAP}\}$. The closure of $\{3, 8, \text{TRAP}\}$ is the set $\{3, 8, 12, 14, 18, \text{TRAP}\}$, since from state 8 there are transitions on external inputs leading to 12 and 14; as well as from 14 to 3 and 18. This a new state in the FSM *EE*. As a result, the FSM *EE* has a transition

$$\{1, 2, 7\}\text{-}u_1/z_1\text{->}\{3, 8, 12, 14, 18, \text{TRAP}\}.$$

Consider now output z_2. $\bigcup_{s\in\{1,2,7\}} h^1_{z_2}(s, u_1) = \{4, 9, \text{TRAP}\}$. The closure of the set $\{4,$

9} is the set {4, 9, 11, FAIL, TRAP}. The presence of state FAIL in the obtained set means that the I/O sequence u_1/z_2 is forbidden. As a result, the FSM *EE* has no transition from the state {1, 2, 7} labeled with u_1/z_2.
Take next input u_2. $\bigcap_{s\in\{1,2,7\}} h^2(s, u_2) = \{z_1, z_2\}$. We have $\bigcup_{s\in\{1,2,7\}} h^1_{z_1}(s, u_2) = \bigcup_{s\in\{1,2,7\}} h^1_{z_2}(s, u_2) = \{\text{TRAP}\}$. Thus, there is a "don't care" transition in the FSM *EE*, namely, {1, 2, 7}-$u_2/z_1,z_2$->{TRAP}. In a similar way we proceed with a newly obtained state {3, 8, 12, 14, 18, TRAP}. The final result, i.e. the FSM *EE*, is shown in Figure 4. As the example shows, the procedure of constructing the embedded equivalent is quite straightforward and we do not elaborate it further to save space for other results.

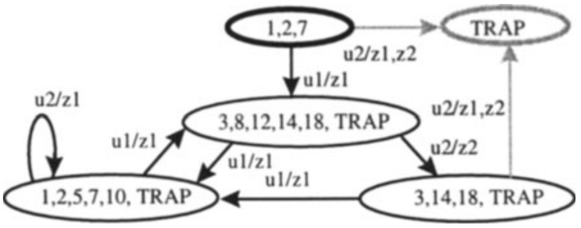

Figure 4 The embedded equivalent *EE*.

The embedded equivalent of the component in context explicitly characterizes all implementations conforming to a given specification *Spec* in context *C*.
Theorem 3.1. Given the specification *Spec* of the component, the context *C*, and an implementation FSM *Imp* over the same alphabets *U* and *Z*, as *Spec*, let *Imp*∘*C* be the composed machine. Then *Imp*∘*C* is equivalent to *RS* = *Spec*∘*C* iff $Imp \leq EE$.
Proof. Let *Imp* be a reduction of *EE*. Suppose that the FSM *Imp*∘*C* is not equivalent to the machine *RS*. Then there exists an external input sequence $\delta\in X^*$ such that *Imp*∘*C* is not equivalent to *RS* w.r.t. this sequence, i.e. the pair β/γ of sequences β and γ that are induced by δ at the inputs of *Imp* and the context *C* is forbidden w.r.t. δ. Thus, the trace β/γ of *Imp* is not an I/O sequence of *EE*; therefore *Imp* is not a reduction of *EE*. A contradiction.

Suppose now that the FSM *Imp*∘*C* is equivalent to *RS* but *Imp* is not a reduction of *EE* w.r.t. an appropriate input sequence β, i.e. the output sequence γ of *Imp* to β is not in the set of output sequences of *EE* to β. Then, by definition of *EE*, there exists a sequence $\delta\in X^*$ such that the trace β/γ is forbidden w.r.t. δ, i.e. the FSMs *Imp*∘*C* and *RS* are not equivalent w.r.t. δ. A contradiction. ❐

We know that a similar characterization of conforming implementations can be obtained based on the most general solution to the equation $G\circ C \cong Spec\circ C$ with *G* being a free variable [PYB96]. As discussed in [PYB96], based on the solution *G* "local" tests to test the component in isolation can be derived, however, these tests are not easy to translate into external tests. Unlike the general solution *G*, the embedded equivalent gives an effective answer to the problem of test translation, as we are about to demonstrate.

Let $EE = (P, U, Z\cup\{fail\}, H, p_0)$. By the definition of the embedded equivalent, each trace in $Tr(EE) = \bigcup_{\beta\in U^*} H^2(p_0,\beta)$ is permissible w.r.t. any external input sequence while for any other trace $\beta/\gamma \in (U/Z)^*\backslash Tr(EE)$ there exists a sequence $\alpha(\beta/\gamma)$ such that β/γ is forbidden w.r.t. $\alpha(\beta/\gamma)$. Consider an arbitrary sequence $\beta \in U^*$. If the set $H^2(p_0,\beta)$ contains all possible output sequences of the same length as β, no sequence $\alpha(\beta/\gamma)$ exists. We use $Z^{|\beta|}$ to denote the set of all sequences in Z^* which have the length of β. Then for each sequence $\gamma\in Z^{|\beta|} \setminus H^2(p_0,\beta)$, a sequence $\alpha(\beta/\gamma)$ exists. Its X-projection is the external test $\alpha(\beta/\gamma)^x$ that can detect an erroneous behavior of the embedded component with the trace β/γ. If now we find at least one sequence $\alpha(\beta/\gamma)$ for each $\gamma\in Z^{|\beta|} \setminus H^2(p_0,\beta)$ and derive the X-projection we have an external test suite which detects all faults internally revealed by the sequence β (in the following section we elaborate a proper method for finding sequences $\alpha(\beta/\gamma)^x$).

At the first sight, the price of this solution seems high since the number of sequences in the set $Z^{|\beta|} \setminus H^2(p_0,\beta)$ is exponential. The following observation helps us to drastically reduce it. Any extension of a forbidden trace (β/γ) is forbidden as well, therefore if we have already found a sequence $\alpha'(\beta'/\gamma')$ for a prefix β' of the sequence β there is no need to consider any extension of β'/γ'. The question comes now how we could choose input sequences $\beta \in U^*$ based on the given embedded equivalent.

Consider the fault model $F = <EE, \leq, \mathfrak{I}_m(U, Z)>$, where $\mathfrak{I}_m(U, Z)$ is the set of all possible implementations with up to m states over the alphabets U and Z, where $m \geq n$, the number of states in the given specification of the embedded component *Spec*. There exists a method for deriving a test suite complete w.r.t. this fault model [PYB96a]. We have the following result.

Theorem 3.2. Given an FSM $EE = (P, U, Z\cup\{fail\}, H, p_0)$, let T be a complete test suite w.r.t. the fault model $<EE, \leq, \mathfrak{I}_m(U, Z)>$. Then the set

$$E = \{\alpha(\beta/\gamma)^x \mid \beta\in T \;\&\; \gamma\in Z^{|\beta|}\setminus H^2(p_0, \beta)\}$$

is a complete test suite w.r.t. the explicit fault model $<RS, \cong, \mathfrak{I}_m(U, Z)\circ C)>$.

Consider our working example. The specification of the embedded component *Spec* has three states (Figure 2). We assume that no fault in the component increases the number of states, i.e. $m = 3$. The method of [PYB96a] applied to the FSM *EE* (Figure 4) produces the following test suite:

$$T=\{u_1u_1u_1u_1u_2;\ u_1u_1u_1u_2u_1u_2;\ u_1u_1u_2u_1u_2;\ u_1u_1u_2u_2u_2;\ u_1u_2u_1u_1u_2;\ u_1u_2u_1u_2u_2\}$$

It is complete w.r.t. the fault model $<EE, \leq, \mathfrak{I}_3(U, Z)>$ and once translated into external sequences (see next section) it is complete w.r.t. the explicit fault model $<RS, \cong, \mathfrak{I}_3(U, Z)\circ C)>$.

In practical situations, we are often ready to sacrifice complete coverage of all output and transition faults for shorter tests. We may consider, for example, the fault model $<EE, \leq, \mathfrak{I}_{Spec}>$, where $\mathfrak{I}_{Spec}$ denotes the set of all FSMs that are mutants of the FSM *Spec* with output faults. In our example, we construct a transition tour of the FSM *EE*: $u_1u_1u_1u_2u_1u_2$ (there is no need to cover any transition to the TRAP state since they are not executable in context). It is just one of the six sequences in the test suite T. Note that this sequence does not cover all the transitions of the original specification of the component *Spec* (Figure 2b). At the same time, not each transition tour of the latter covers all the transitions of the former.

4 TRANSLATION OF INTERNAL TESTS INTO EXTERNAL TESTS

Once the embedded equivalent of a given component in context is constructed, an internal test suite could be produced based on a chosen fault model (e.g. output or transition faults). Tests are internal and should be translated into external tests which could be applied to the context. Theorem 3.2 suggests how this could be done. Let $\beta \in U^*$ be an internal input sequence, i.e. internal test. Applied to the FSM $EE = (P, U, Z \cup \{fail\}, H, p_0)$ the sequence β produces the set of output sequences $H^2(p_0, \beta)$. Each trace β/δ such that $\delta \in H^2(p_0, \beta)$ is permissible w.r.t. any external input sequence. At the same time for any trace β/γ such that $\gamma \in Z^{|\beta|} \setminus H^2(p_0, \beta)$, there exists a sequence $\alpha(\beta/\gamma)$ such that the trace β/γ is forbidden w.r.t. $\alpha(\beta/\gamma)^x$. Once found, the sequence $\alpha(\beta/\gamma)^x$ is an external test which forces the context to execute the internal test β provided that an IUT executes the trace β/γ. If we find a sequence $\alpha(\beta/\gamma)^x$ for all $\gamma \in Z^{|\beta|} \setminus H^2(p_0, \beta)$ then we have a set of external tests corresponding to a single internal test β. To execute the internal test β against a particular implementation of the component one external test suffices, but since we do not know much about an IUT we should use all of them.

The key issue is then to find for a given test $\beta \in U^*$ all (or at least one) sequences $\alpha(\beta/\gamma)$ for every trace β/γ, where $\gamma \in Z^{|\beta|} \setminus H^2(p_0, \beta)$. This could actually be solved by constructing a synchronous product of the approximation and an FSM representing all the forbidden traces β/γ. The approach is very similar to that of finding from a specification a test covering a given test purpose for testing an isolated implementation, see for example, [FJJV96]. In that approach, test derivation is based on a depth-first traversal of the product. In our case, a forbidden trace serves as a test purpose and the approximation $[[Spec]]_C$ with two distinct types of alphabets X and U plays the role of a specification. The procedure is thus slightly more involved:

1. We construct an FSM, called a *test machine* $A(\beta)$, such that has a distinct state for each trace α/δ, where α is a proper prefix of β and $\delta \in H^2(p_0, \alpha)$, as well as two designated states FAIL and TRAP. Transitions are defined in the following way. For each trace $\alpha u/\delta z$ such that αu is a prefix of β and $\delta z \notin H^2(p_0, \alpha u)$, we define a transition from a corresponding state to the FAIL state labeled with u/z. The FAIL state has looping transitions labeled with all pairs u/z, $u \in U$ and $z \in Z$. All traces β/δ, where $\delta \in H^2(p_0, \beta)$, take the test machine to the TRAP state. Since each state of the test machine accepts at most one input u of the sequence β, we define transitions to the TRAP state for all the remaining internal inputs in U and all internal outputs in Z. Once the TRAP state is reached, no forbidden trace can be found for the internal

test β. To synchronize the test machine with the approximation we should also equalize their alphabets. In particular, we augment the test machine with the external inputs in X, they cause looping transitions at each state with the *null* output.

2. Given the two FSMs, $A(\beta) = (Q, X \cup U, Z \cup \{null\}, g, q_0)$ and $[[Spec]]_C = (S, X \cup U, Z \cup \{fail, null\}, h, s_0)$ we are interested in input sequences that simultaneously lead them to the FAIL states. In the former machine such a sequence causes a forbidden trace, while in the latter, its x-projection is the external test for this forbidden trace. We construct the synchronous product of $A(\beta)$ and $[[Spec]]_C$ as an FSM $A(\beta) \times [[Spec]]_C = (Q \times S, X \cup U, Z \cup \{fail, null\}, g \times h, q_0 s_0)$, where $g \times h(qs, a) = \{ [(q_b^1(q, a), h_b^1(s, a)), b] \mid b \in g^2(q, a) \cap h^2(s, a) \}$ if $g^2(q, a) \cap h^2(s, a) \neq \emptyset$ and $g \times h(qs, a) = \{(\text{FAIL}, \text{FAIL}), fail\}$ if $g^2(q, a) \cap h^2(s, a) = \emptyset$. By definition of $A(\beta)$, $g^2(q, a) \cap h^2(s, a) = \emptyset$ implies $a \in X$, $g^2(q, a) = \{null\}$, and $h^2(s, a) = \{fail\}$.

3. We find all traces of the synchronous product $A(\beta) \times [[Spec]]_C$ from the initial global state to the global state (FAIL, FAIL). To shorten the length of a trace we could skip looping transitions finding shortest paths from the initial state.

4. For a particular forbidden trace β/γ different sequences $\alpha(\beta/\gamma)$ may be found, it is sufficient to choose one of them for each forbidden trace. In other words, for each forbidden trace β/γ, we find one trace α/β with the $(U \cup Z)$-projection β/γ that takes the FSM from its initial state to the state (FAIL, FAIL). Alternatively, we could optimize the number of external tests by solving a set cover problem [John74]. Finally, we find the x-projection of the obtained sequences.

We illustrate the construction using our example. Consider, as an example, the internal test u_1u_1. Figure 5 shows the test machine constructed for this test. There are two forbidden traces, u_1/z_2 and $u_1/z_1u_1/z_2$, the test machine enters the FAIL state after these traces.

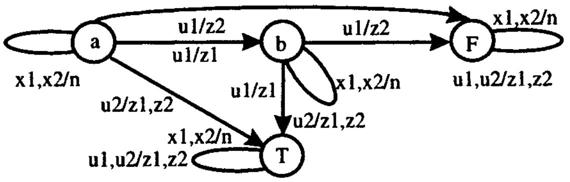

Figure 5 The test machine derived from the input sequence u_1u_1.

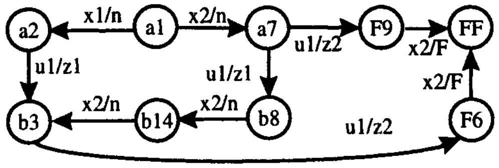

Figure 6 A fragment of the machine $A(\beta) \times [[Spec]]_C$ for $\beta = u_1u_1$.

A fragment of the product of the test machine and approximation (Figure 3) which contains the necessary traces is shown in Figure 6. For the forbidden trace u_1/z_2 we have a single sequence $x_2u_1x_2$, and for $u_1/z_1u_1/z_2$ we have two sequences $x_1u_1u_1x_2$ and $x_2u_1x_2x_2u_1x_2$ reaching the state (FAIL, FAIL). Accordingly, there are two possible solutions $\{x_2x_2, x_1x_2\}$ and $\{x_2x_2, x_2x_2x_2x_2\} = \{x_2x_2x_2x_2\}$. To execute the internal test u_1u_1 we may use a single external test $x_2x_2x_2x_2$ or two tests x_2x_2 and x_1x_2.

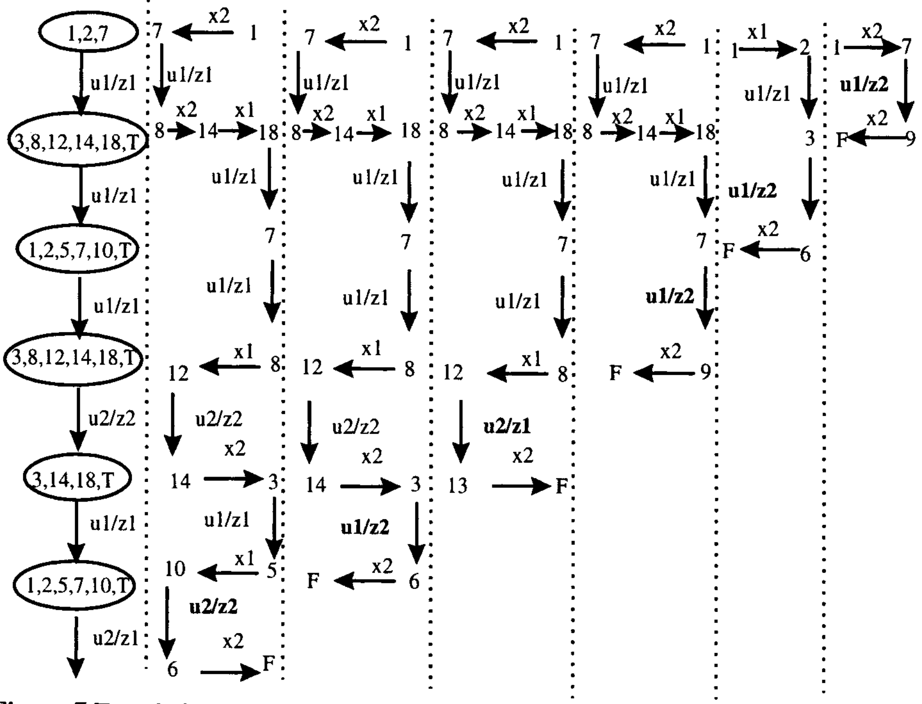

Figure 7 Translating the internal test $u_1u_1u_1u_2u_1u_2$ into external tests.

The above procedure allows us to find not only a sequence $\alpha(\beta/\gamma)^x$ for each forbidden trace β/γ but also to translate a given internal test β into a number of external tests by deriving them for each forbidden trace which an IUT can execute when the test β is internally applied. It facilitates the optimization of the number of external tests since it can deliver the set of all minimal external tests for each internal test. However, the approach relies on the product of the approximation and test machine which may have too many states to be even constructed. To reduce the complexity of test translation we need a simpler method for finding a single sequence $\alpha(\beta/\gamma)$ for a given trace β/γ and not all of them.

The idea of such a method is based on the fact that states of the embedded equivalent constructed as subsets of states of the approximation allow us to backtrack a sequence $\alpha(\beta/\gamma)$ for a given trace β/γ starting from the final state FAIL in the approximation. The backtracking procedure is illustrated in Figure 7 for the transition tour $u_1u_1u_1u_2u_1u_2$ of the embedded equivalent (Figure 4). The transition graph presents transitions in the embedded equivalent caused by this internal test. The columns correspond to all forbidden traces the test can cause in an IUT. A forbidden part u/z of each trace leading eventually the FAIL state (F) is depicted in bold. Consider the longest forbidden trace $u_1/z_1u_1/z_1u_1/z_1u_2/z_2u_1/z_1u_2/z_2$. The suffix u_2/z_2 is executed in the approximation from one of the states {1,2,5,7,10, TRAP}. By direct inspection of Figure 3 we find that it is state 10. FAIL state is reached from state 10 with u_2/z_2x_2 through state 6. Backtracking continues until the initial state 1 is reached. The x-projection gives the test $x_2x_2x_1x_1x_2x_1x_2$ for the considered forbidden trace.

As a result, to execute a single internal test, a transition tour of the FSM *EE* $u_1u_1u_1u_2u_1u_2$, the following external input sequences are required:

$$\{x_1x_2;\ x_2x_2x_1x_2,\ x_2x_2x_1x_1x_2x_2;\ x_2x_2x_1x_1x_2x_1x_2\}.$$

Four sequences of the total length of 19 external test events are needed to detect all output faults in the embedded component.

Next we apply the backtracking procedure to the internal test suite T complete w.r.t. the fault model $<EE, \leq, \mathfrak{I}_3(U, Z)>$ (see Section 4) and obtain the following external test suite complete w.r.t. the explicit fault model $<RS, \cong, \mathfrak{I}_3(U, Z)\circ C)>$:

$$\{x_1x_1x_1x_1x_1x_2;\ x_1x_1x_1x_2;\ x_1x_1x_2x_2x_1x_2;\ x_1x_1x_2x_2x_2;\ x_1x_2x_1x_1x_2;\ x_1x_2x_1x_2;\ x_2x_2x_1x_1x_2x_1x_2;$$
$$x_2x_2x_1x_1x_2x_2;\ x_2x_1x_1x_1x_2;\ x_2x_1x_1x_2;\ x_2x_1x_2x_1x_1x_2;\ x_2x_1x_2x_2x_2;\ x_2x_2x_1x_2\}.$$

The total length is 67, as is indicated in [PYB96], where such a test suite was obtained by an ad hoc procedure. Note that in this particular example, translation of a complete internal test suite into an external one almost doubles the length of tests. The increase depends, of course, on how "transparent" is the context to signals from/to the embedded component.

5 CONCLUSION

In this paper, we have considered the problem of test derivation aimed at detecting faults in a component embedded within a given system modeled by communicating state machines assuming that the rest of the system has no faults. The presented results are based on a general framework for testing in context elaborated in the previous work [PYBD96], [PYB96], [PYD94], [PYLD93]. We have demonstrated that tests which detect all predefined (transition or output) faults can be systematically derived through the following steps. First, we construct a so called approximation of the component in context, which characterizes the behavior of any implementation of the component. This step was elaborated in our previous papers. New procedures proposed in this paper are as follows. The approximation is transformed into an embedded equivalent of the component. The latter contains the behavior of any conforming implementation and is used to derive internal tests complete with respect to a chosen fault model. An existing method for deriving tests from a nondeterministic FSM and reduction relation between an implementation and its specification can be applied at this point. Since we assume that no access is possible to the embedded component internal tests have to be translated into external tests applied at available test access points. Two approaches have been elaborated to solve the last problem. Compared to the published results, we have elaborated a systematic approach which leads to better results, i.e. shorter tests with the same fault coverage guarantee.

Possible future work is related to generalization of this approach to nondeterministic communicating state machines and extended finite state machines. It would also be interesting to see whether the constructions used in our approach could be further simplified to treat real-size specifications. More research is required to merge the two approaches, the one elaborated in this work and the other based on a partial exploration of a composed machine while preserving their advantages.

Acknowledgments. This work was partially supported by the NSERC grants OGP0194381 and STRGP200.

6 REFERENCES

[ABBD95] Aziz, A., Balarin, F., Brayton, R. K., DiBenedetto M. D., and Saldanha, A. (1995) Supervisory control of finite state machines, *Proceedings of the 7th International Conference CAV'95*, pp. 279-292.

[Boch78] Bochmann, G. v. (1978) Finite state descriptions of communication protocols, *Computer Networks*, 2.

[BrZa83] Brand, D., and Zafiropulo, P. (1983) On communicating finite state machines. *Journal of ACM*, **30**, **2**, 323-42.

[FJJV96] Fernandez, J. C., Jard, C., Jeron, T., and Viho, G. (1996) Using on-the-fly verification techniques for the generation of test suites, *Proceedings of the 8th International Conference CAV'96*.

[HeBr95] Heerink, L. and Brinksma, E. (1995) Validation in context, *Proceedings of the 15th IFIP International Symposium on Protocol Specification, Testing, and Verification*, Chapman & Hall.

[HLS96] Huang, S., Lee, D., and Staskauskas, M. (1996) Validation-based test sequence generation for networks of extended finite state machines, *the Proceedings of the IFIP 1st Joint International Conference FORTE/PSTV*, Chapman & Hall, pp. 403-418.

[HoUl79] Hopcroft, J. E., and Ullman J. D. (1979) *Introduction to automata theory, languages, and computation*, Addison-Wesley, New York.

[John74] Johnson, D.S. (1974) Approximation algorithms for combinatorial problems. *Journal Comput. Syst. Sci.*, 9, pp. 256-78.

[KoTa95] Koppol, P. V., Tai, K. C. (1995) Conformance testing of protocols specified as labeled transition systems, *Proceedings of the 8th International Workshop on Protocol Test Systems (IWPTS'95)*, pp. 143-158.

[LBP94] Luo, G., Bochmann, G. v., and Petrenko, A. (1994) Test selection based on communicating nondeterministic finite state machines using a generalized Wp-method. *IEEE Trans. on Soft. Eng.*, SE-20, 2, 149-62.

[LJK95] Lin, B., de Jong G., and Kolks, T. (1995) Hierarchical optimization of asynchronous circuits, *Proceedings of the 32nd DAC*, pp. 712-717.

[LSKP96] Lee, D., Sabnani, K. K., Kristol, D. M., and Paul S. (1996) Conformance testing of protocols specified as communicating finite state machines - a guided random walk based approach, *IEEE Trans. on Communication*, vol. 44, 5.

[MeBo83] Merlin, P., and Bochmann, G. v. (1983) On the construction of submodule specifications and communication protocols, *ACM Trans. on Programming Languages and Systems*, Vol. 5, No. 1, pp. 1-25.

[PBD93] Petrenko, A., Bochmann, G. v., and Dssouli, R. (1993) Conformance relations and test derivation, Invited Paper, *Proceedings of the 6th International Workshop on Protocol Test Systems (IWPTS'93)*, pp.157-178.

[PYBD96] Petrenko, A., Yevtushenko, N., Bochmann, G. v., and Dssouli, R. (1996) Testing in context: framework and test derivation, *Computer Communications Journal*, Special issue on Protocol Engineering, 19, pp.1236-1249.

[PYB96] Petrenko, A., Yevtushenko, N., and Bochmann, G. v. (1996) Fault models for testing in context, *Proceedings of the IFIP 1st Joint International Conference FORTE/PSTV*, Chapman & Hall, pp. 163-178.

[PYB96a] Petrenko, A., Yevtushenko, N., and Bochmann, G. v. (1996) Testing deterministic implementations from nondeterministic fsm specifications, *Proceedings of the 9th IWTCS'96*, Chapman & Hall, pp.125-140.

[PYD94] Petrenko, A., Yevtushenko, N., and Dssouli, R. (1994) Testing strategies for communicating fsms, *Proceedings of the 7th IWTCS'94*, pp. 193-208.

[PYLD93] Petrenko, A., Yevtushenko, N., Lebedev, A., and Das, A. (1993) Nondeterministic state machines in protocol conformance testing, *Proceedings of the 6th IWPTS*, pp. 363-378.

[QiLe91] Qin, H., and Lewis, P.(1991) Factorization of finite state machines under strong and observational equivalencies, *Journal of Formal Aspects of Computing*, Vol. 3, pp. 284-307.

[Star72] Starke, P.H. (1972) *Abstract automata*. North-Holland/American Elsevier.

[SiLe89] Sidhu, D. P., and Leung, T. K. (1989) Formal methods for protocol testing: a detailed study, *IEEE Trans. on Soft. Eng.*, SE-15, 4, pp.413-426.

[West86] West, C. (1986) Protocol validation by random state exploration, *Proceedings of the 6th ISPSTV.*

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

7 BIOGRAPHY

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

Nina Yevtushenko received the 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.