(S. Budkowski and E. Najm, eds, Protocol Engineering: Part 2, vol. 55(1-2) of Annals of Telecommunications (special issue), Jan-Feb 2000)

Conformance Testing in Systems with Semicontrollable Interfaces*

Mariusz A. Fecko[1], M. Ümit Uyar[2,†], Paul D. Amer[1], Adarshpal S. Sethi[1]

[1] Computer and Information Sciences Department
University of Delaware, Newark, DE

[2] Electrical Engineering Department
City College of the City University of New York, NY

Abstract

In a conformance testing environment, an implementation under test (IUT) communicates with multiple entities. A tester may have differing degrees of control on the interactions between these entities and the IUT: directly controllable, semicontrollable, or uncontrollable. Semicontrollable inputs most likely render portions of an IUT untestable. In addition, multiple communicating entities may create race conditions during testing. This paper presents a test generation methodology for the systems where the semicontrollable inputs can be generated indirectly. The test sequences obtained from the converted graph fully utilize the semicontrollable inputs (where possible) while avoiding the race conditions. Although, for the most general case, the graph conversion results in an exponentially large number of nodes, practical considerations make the converted graph size feasible. This approach is used to generate tests for MIL-STD 188-220B. By applying the proposed graph conversion and the race condition elimination techniques, the number of testable state transitions increased from approximately 200 to over 700, which represents a coverage of 95% of the transitions defined in the specification.

Keywords: Conformance testing; Embedded testing; Test case generation; Communication protocol specification and testing

1 Introduction

In the automated generation of conformance tests based on the formal description of a protocol [1, 5, 16, 20, 21, 29, 33, 34], one significant problem is taking into account a tester's limited controllability on generating inputs to an Implementation Under Test (IUT) [20, 28]. This limited control almost always renders certain protocol features untestable.

In an embedded testing environment, a composite System Under Test (SUT) [20, 28] consists of two parts: (1) an IUT embedded within the SUT (often referred to as the *test component* [37]), and (2) all entities within the SUT other than the IUT (referred to as the *test context* [25, 35, 37]). A reasonable assumption adopted by most researchers and practitioners about the test context is that it is fault-free and therefore testing should focus on the IUT [35, 37]. The compliance of the IUT to its specification can only be verified from the global system behavior by examining the SUT's input and output events [13].

In a practical embedded testing environment [28], where an IUT communicates with the test context consisting of multiple entities, a tester may have differing degrees of controllability on the interactions between these entities and the IUT [19, 13]. It may not be possible for a tester to directly apply some of the inputs defined in the finite-state machine (FSM) model of an IUT; similarly, a tester may be unable to

*This work is supported by ARO SPP administered by Battelle (DAAL03-91-C-0034), by ARO (DAAL03-91-G-0086), and by ATIRP Consortium sponsored by the ARL under the FedLab Program (DAAL01-96-2-0002).

†Dr. Uyar initiated this research while a Visiting Associate Professor at University of Delaware.

observe some of the outputs generated by the IUT. Those interactions that are not directly controllable are most likely to introduce nondeterminism and/or race conditions during testing, leaving certain portions of an IUT untestable.

Oftentimes, within an SUT, an IUT communicates with the test context's entity(s) FSM_i that is not directly controllable (i.e., the tester cannot directly apply inputs to FSM_i). Consider an input $a_{i,j}$ that cannot be directly applied to the IUT by a tester, but can be generated as an output by FSM_i. In some cases, it may be possible to force FSM_i to generate $a_{i,j}$ to the IUT. If the tester applies an appropriate input to the IUT, which then generates an input to FSM_i, which in turn generates $a_{i,j}$ as input back to the IUT, some of the interfaces become semicontrollable (as opposed to uncontrollable).

In addition, in an embedded environment where an IUT interacts with multiple entities, race conditions and/or nondeterminism can occur during testing. If an IUT moves into a state in which several inputs from different interfaces are waiting to be processed, choosing which input is consumed first may be a nondeterministic decision of the IUT.

Within the framework of embedded testing [28], uncontrollable events are discussed by Phalippou [26] and Cavalli et al. [5]. Several approaches such as ferry clip [36] and astride responder [27] were suggested to address the limited controllability over an IUT. However, such approaches require a tester-designed entity within the SUT, which limits their applicability. Recently, Petrenko et al. [24, 25], Lima and Cavalli [18], and Yevtushenko et al. [35] focus on embedded test generation based on fault models. For large communication protocols, it may be difficult to find detailed fault classes.

As opposed to the fault model oriented methods, the approach used in this paper does not attempt to guarantee full fault coverage. Instead, this paper provides a practical algorithmic technique for test generation that utilizes as many indirectly controllable inputs as possible without creating nondeterministic behavior of the test system. A transition graph is built for test derivation without explicitly constructing a composite global FSM for an SUT. For each FSM_i, only those transitions that can be used to test IUT's transitions are considered during the graph construction. This approach has a significant advantage, since an FSM_i's state and transition space may be prohibitively large. For the worst case, all transitions of all the FSMs communicating with the IUT may be involved into the composition. However, based on the practical experience with several protocols [7, 2], for any given semicontrollable input, only a small portion of a communicating FSM is required to generate the semicontrollable inputs. An analysis of a system model size and the length of the test sequence with respect to the SUT parameters is presented. By controlling the model size, this technique could be applied to large communications protocols.

A sketch of the algorithm to build the transition graph used in this paper was introduced in [11] (full algorithm and formal analysis of its time complexity is available in [10]). This paper extends that work by considering controllability of an FSM_i associated with a semicontrollable interface. This paper enhances the model by including the preambles and postambles to move the IUT and the FSM_i together into a desired state.

MIL-STD 188-220B Data Link Layer [7] and the IEEE 802.2 LLC Connection Component [2] are considered as real-life examples of protocols that possess either semicontrollable inputs. In MIL-STD 188-220B [7], over 70% of the transitions cannot be directly controlled. The initial results of applying the method introduced in this paper to MIL-STD 188-220B to generate conformance tests are promising: the number of testable transitions increased to over 700 from approximately 200 for the Class A–Type 1 Service Datalink module [7, 8].

The test generation approach presented in this paper studies embedded testing for an environment with a one-party lower tester. Extension of this work will cover the systems with multiple testers, which requires addressing synchronization issues in multi-party testing [34].

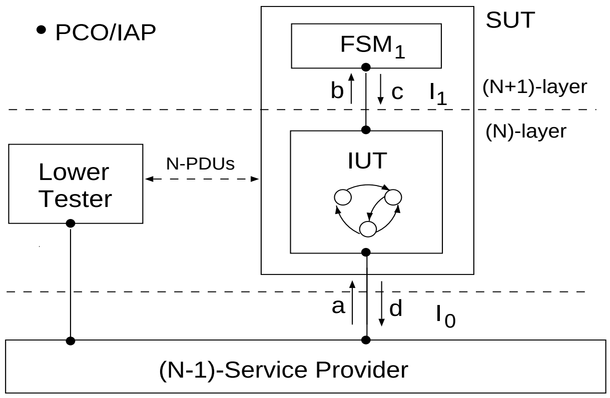

Figure 1: Testing (N)-layer IUT with an (N+1)-layer semicontrollable interface.

This paper is organized as follows. Section 2 presents a formal definition of the *controllability problem* and provides examples from real-life protocols. Some related work is discussed in Section 3. Section 4 defines a system model for a testing environment with multiple interfaces with different degrees of controllability. Practical issues are introduced into this model in Section 5. In Section 6, the application of the graph conversion algorithm and minimum cost test sequence generation techniques is presented. Controlling FSMs associated with semicontrollable interfaces is discussed in Section 7.

2 Problem definition

Consider a testing environment shown in Figure 1. The SUT contains an IUT (test component), which interacts with FSM_1 (test context). FSM_1, implemented inside the SUT, interacts with the IUT through interface I_1. Since FSM_1 represents a well-defined part of the SUT, it is reasonable to assume that the specification for FSM_1 is available. The points at which a testing system can apply inputs to and observe outputs from the IUT are called *points of control and observation* (PCOs) [13, 19]. Each of the IUT's interfaces is associated with a full-duplex PCO through which inputs and outputs can be exchanged. If a PCO of an IUT is not accessible within an SUT, this PCO is called an Implementation Access Point (IAP) [14, 37]. As introduced in [11], each input can be one of three different types: (1) **directly controllable:** a tester can directly apply the input to the IUT through the PCO; (2) **semicontrollable:** a tester cannot directly apply the input to the IUT through the PCO (or IAP). However, it is possible to utilize one of the FSMs interacting with the IUT to supply this input indirectly; and (3) **uncontrollable:** the input may be supplied through a PCO (or an IAP) without any explicit action of the tester. This means that the input may be generated in the testing system without the tester's control.

If a PCO (or IAP) has any semicontrollable inputs and does not have any uncontrollable inputs, we say that its associated interface and the FSM are semicontrollable. If there are no semicontrollable or uncontrollable inputs, the interface and the FSM are called directly controllable. In this paper, we consider that each interface has only one type of input: either directly controllable or semicontrollable. The analysis can be easily adopted to the case where an interface has a combination of directly controllable and semicontrollable inputs. The uncontrollable inputs are not considered in this paper. Typically, a lower tester (LT) FSM [19, 13] has a directly controllable interface. A timer FSM, whose only inputs come from an IUT (e.g., start, restart, and stop the timer), has a semicontrollable interface.

Consider the testing framework in Figure 1. Since the interface I_1 is not exposed in the SUT, the tester can neither directly apply inputs nor observe the outputs between the IUT and the (N+1)-layer. Therefore, at best the interface I_1 is semicontrollable, provided that FSM_1 can be utilized to supply inputs to the IUT. On the other hand, the tester can apply inputs to the IUT directly at interface I_0 by using an LT, which exchanges N-PDUs with the IUT by using the (N-1)-Service Provider. The interface I_0 is therefore directly controllable through the (N-1)-Service Provider, whose erroneous behavior may or may not be observed as an error of the IUT, whether or not the actual error in the IUT occurred.

To test an IUT's transition from state v_i to v_j, the following steps must be taken: (1) put an IUT into state v_i, (2) apply required input and compare the output(s) generated with those defined in the specification, and (3) (optionally) verify that the new state of the FSM is v_j.

If the IUT's transitions are triggered by the inputs from a semicontrollable interface I_1, the tester must use one of the directly controllable interfaces to force the IUT to generate outputs to I_1. These outputs are applied to FSM_1 at I_1's PCO. As response, FSM_1 will send back inputs to the IUT through semicontrollable interface I_1. These inputs will trigger the desired transitions to be tested in the IUT.

This paper addresses the problem of generating realizable test sequences for a fault-free IUT communicating with a fault-free context through multiple semicontrollable interfaces. By executing the generated test sequences, we expect the errors in the IUT to be uncovered based on performing steps (1) through (3) above for each transition in a test sequence. The problem of test generation as defined above will be referred to as the *controllability problem.*

2.1 Practical examples

The controllability problem was motivated by two real protocols implementations in which certain transitions within an (N)-layer IUT can be tested only by utilizing an SUT's (N+1)-layer. Moreover, there is a danger of introducing race conditions to test sequences while utilizing an (N+1)-layer indirectly.

MIL-STD 188-220B [7] is a military standard for interoperability of command, control, communications, computers, and intelligence over Combat Net Radios. There are many transitions in 188-220B that cause controllability problems during testing. Without these transitions, over 70% of the transitions cannot be tested because many protocol states are unreachable, 70% of the transitions cannot be tested because of these semicontrollable inputs. In this case, test coverage is seriously reduced. However, by applying the technique introduced in this paper, almost all (>95%) transitions defined in the specification can be tested (the number of testable transitions rose to over 700 from approximately 200 for the Class A–Type 1 Service Datalink module [7, 8]). The application of this paper's methodology to MIL-STD 188-220B is detailed in [9].

In the **IEEE 802.2 LLC Type 2 Connection Component** [2], the LLC layer IUT communicating with a semicontrollable upper layer is considered. Suppose that, when an IUT is in state v_p, the tester applies from the lower tester a sequence of inputs that causes FSM_i to supply input $a_{i,j}$ to the IUT. The tester desires this input to be consumed at state v_q. In general, a race condition will occur if this input from FSM_i is consumed by the IUT before the IUT reaches state v_q. A valid test sequence should avoid these conditions while traversing the IUT's transitions. A detailed scenario that shows how race conditions may occur in a test sequence for the 802.2 LLC's IUT is presented in [11].

3 Related work

The controllability problem defined in Section 2 is related to a protocol and software engineering issue of testing embedded systems [28, 31]. An IUT is *embedded* in the SUT (Figure 1). The SUT is directly accessible by a tester, whereas the IUT can be tested only as a component embedded in the SUT.

Rayner [28] discusses the concept of testing an IUT embedded in a multilayer SUT (with an upper tester available above the SUT) for various OSI test methods [19] in which an (N)-layer IUT has only two interfaces: one controllable with a lower tester, and one semicontrollable with the (N+1)-layer. A similar model, but limited to a single layer embedded testing, is presented by Timohovich [31]. A test sequence is derived based on a combined finite automaton, which is constructed using an IUT's FSM and a simplified description of the adjacent layers. A need to introduce mechanisms to control uncontrollable events, a necessity to avoid nondeterminism in test sequences, and the limits imposed by test architecture on the controllability and observability of an IUT are discussed in [5, 26]. Compared to the models in [27, 31, 36], the test system considered in this paper (Figure 2) is a more general architecture, where an IUT has interfaces with multiple (N)- and (N+1)-layer entities, and there is no upper tester available.

In the test system depicted in Figure 1, FSM_1 is part of the protocol defined by the protocol standard. Therefore, the tester cannot redefine the states and transitions of FSM_1 suitable to her needs. There are several testing frameworks in which, as in Figure 1, the interface between an IUT and FSM_1 resides inside the SUT, but FSM_1 is defined by the tester. One of these frameworks is the ferry clip testing method [36], where the SUT contains an entity called a passive ferry clip to apply inputs to the IUT. In such a test system, there exists a special-purpose protocol for exchanging PDUs between an active and a passive ferry clip. Ferry clip testing cannot be directly used in the test system considered in this paper, since the interfaces between the IUT and other entities inside the SUT are not accessible. A related approach [27] uses the "astride responder" to supply inputs to the IUT at the interface I_1. The astride testing is not directly applicable to the system in Figure 1, because there is no tester-designed responder, nor are extra communication channels available.

Recent research has focused on testing embedded components [18, 23, 24, 25, 35, 37]. Petrenko et al. [25] provide a basic framework for "testing in context" based on the model of communicating FSMs. The proposed solution consists in computing a so-called approximation of the specification in context, i.e., the FSM model of the component's properties that can be controlled and observed through the context. The IUT's transfer and output faults are translated into faults of the composite FSM representing an SUT, resulting in tests with guaranteed fault coverage and executability.

Lima and Cavalli [18] propose an approach based on combining the component and the context into a composite machine. The composite machine's transitions that are not affected by the component are called *redundant*; the remaining transitions are called *suspicious*. In the case of a fault-free context, the test sequences traversing only redundant transitions are superfluous. A method is introduced for detecting redundant transitions and sufficient conditions for removing superfluous test cases are given. An extension of this work presented by Yevtushenko et al. [35] contains a rigorous analysis of suspicious transitions, where conditions are provided to detect all of the redundant transitions. The evaluation of test suites for embedded system testing is provided by Zhu et al. [37].

These proposed approaches [18, 23, 24, 25, 35, 37] focus on defining fault models, generating complete test suites (see [32] for a formal definition of a test suite's completeness) and their evaluation with respect to given fault models. However, in many complex protocols detailed fault classes are unknown or are difficult to construct. A systematic approach to finding fault classes that is feasible for large protocols is not given. It is well known that for a specification of $|S|$ states, $|I|$ inputs, and $|O|$ outputs, there exist $((|S| * |O|)^{|S|*|I|} - 1)$ faulty implementations [29]. Therefore, building complete test suites, i.e., the ones guaranteeing full coverage of all faults within the defined fault model [24], for a large number of potential fault models is likely to be impractical due to a prohibitive growth of test suite size. It is unclear how the fault-oriented approaches for testing in context scale with respect to a protocol's size, as detailed analysis of the algorithms' running time and the generated test suite size are not provided.

The approach used in this paper does not attempt to guarantee full fault coverage, since such a goal is unlikely to be achieved with limiting a test suite to a reasonable size. Instead, the emphasis is put on

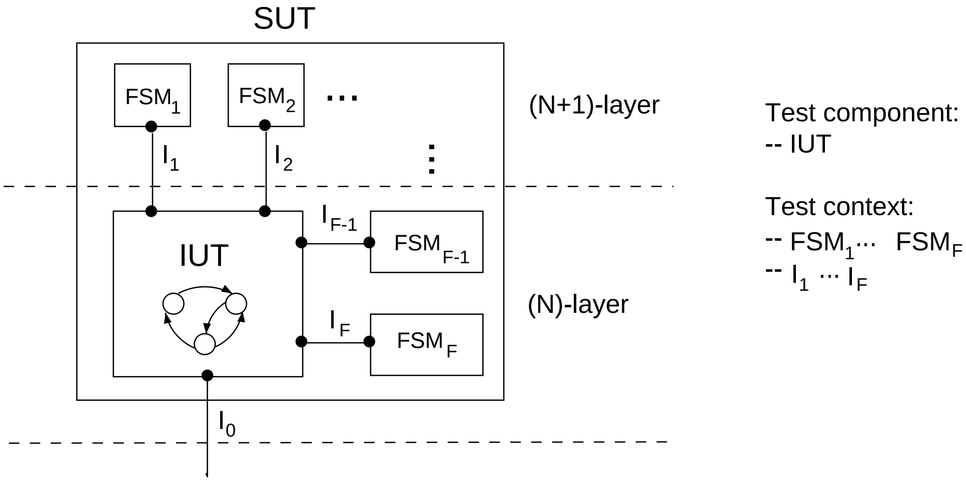

Figure 2: Testing IUT with multiple interfaces

providing the tester with a practical algorithmic technique that achieves a test purpose of utilizing as many indirectly controllable inputs as possible without creating nondeterministic behavior of the test system. Another contribution of the presented methodology is the consideration of race conditions and their avoidance in test generation. In addition, special importance is given to carefully analyzing the growth of a system model and the length of the test sequence with respect to the SUT parameters. By keeping the test sequence length under control, the presented technique could be applied to large communications protocols.

4 Building test system model taking into account controllability problem

In this paper, an FSM model, which is sufficient to model protocols with finite state space and deterministic behavior, is used to represent an implementation.

Consider a testing environment shown in Figure 2. The SUT contains an IUT (test component), which interacts with F FSMs. $FSM_1, \cdots, FSM_F$, implemented inside the SUT, interact with the IUT through semicontrollable interfaces $I_1, \cdots, I_F$, which, together with F FSMs, represent the test context. The goal of test generation in this environment is *to derive a set of tests exercising each transition in an IUT's FSM at least once.* Specifically, given a graph G representing an IUT's FSM, we want to find a minimum cost tour of G such that each transition is covered at least once.

A preliminary version of the model presented in this section was introduced in [11]. This model has been extended here to handle the controllability of semicontrollable $FSM_1, \cdots, FSM_F$.

4.1 System model for IUT with semicontrollable interfaces

Given a graph $G(V, E)$ representing an FSM model of an IUT with multiple semicontrollable interfaces, let us define the following parameters: **(1)** $|V|$—number of nodes in G; **(2)** F—number of semicontrollable interfaces interacting with the IUT; **(3)** $T_i \subset E$—subset of edges in G triggered by the inputs from the i-th semicontrollable interface; **(4)** b_i—buffer size (max. number of inputs buffered) at the i-the semicontrollable interface I_i; **(5)** A_i—set of inputs triggering transitions in T_i; **(6)** O_i—set of outputs of the IUT that are consumed by the semicontrollable FSM_i; **(7)** c_i—number of different transition classes

Edge name	*Input from*	*Output to*	*Edge name*	*Input from*	*Output to*
e1	$LT?x_1$	$FSM_1!o_{1,1}$	e6	$LT?x_6$	$LT!y_6$
e2	$LT?x_2$	$FSM_2!o_{2,1}$	e7	$LT?x_7$	$LT!y_7$
e3	$FSM_1?a_{1,1}$	$LT!y_3$	e8	$FSM_1?a_{1,2}$	$LT!y_8$
e4	$FSM_2?a_{2,1}$	$FSM_1!o_{1,2}$	e9	$LT?x_9$	$LT!y_9$
e5	$LT?x_5$	$FSM_2!o_{2,2}$	e10	$LT?x_{10}$	$LT!y_{10}$

Table 1: Inputs and outputs for the edges of Figure 3. $A?x$ denotes receiving input x from A. $B!y$ denotes sending output y to B.

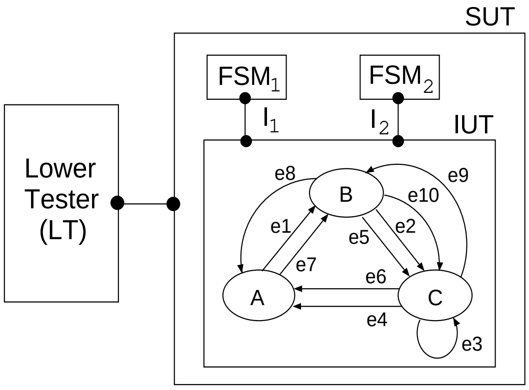

Figure 3: IUT interacting with two semicontrollable interfaces.

in the IUT triggered by inputs at I_i. Two transitions t_1 and t_2 belong to the same transition class $T_{i,j} \subset T_i$ iff they are both made fireable by the same input $a_{i,j} \in A_i$; **(8)** $U_{i,j} \subset E$—set of transitions in the IUT with output $o_{i,j}$ such that, in response to $o_{i,j}$, an input $a_{i,j} \in A_i$ is buffered at I_i; **(9)** $W_{i,j} \subset E$—set of transitions in the IUT with output $o_{i,j}$ such that, in response to $o_{i,j}$, no output is generated by FSM_i.

Let $A_i = \{a_{i,1}, \ldots, a_{i,c_i}\}$ and $O_i = \{o_{i,1}, \ldots, o_{i,m_i}\}$. Let the sets of T_i and U_i be defined as follows: $T_i \stackrel{def}{=} \bigcup_{j=1}^{c_i} T_{i,j}$, and $U_i \stackrel{def}{=} \bigcup_{j=1}^{c_i} U_{i,j}$. Note that there may be several outputs in set O_i that force input $a_{i,j}$ to be buffered at I_i. For the sake of simplicity, let $o_{i,j}$ denote any output forcing $a_{i,j}$ at I_i.

Based on the above definitions, the transitions triggered by the inputs from the semicontrollable interface I_i are divided into c_i classes, each corresponding to a distinct input that fires any transition within the class. No single transition can belong to more than one $T_{i,j}$. Similarly, each transition can belong to only one $U_{i,j}$. In general, $T_{i,j}$ and $U_{i,j}$ may or may not be disjoint.

Example : *Consider the IUT of Figure 3 which is interacting with FSM_1 and FSM_2 through semicontrollable interfaces I_1 and I_2, respectively. The IUT's FSM is described in Table 1. Transition $e1$, triggered by input x_1 from an LT, generates output $o_{1,1}$ to FSM_1. In response, FSM_1 sends back input $a_{1,1}$ which triggers transition $e3$. (By definition, $a_{i,j}$ is the expected response to $o_{i,j}$.) Transition $e2$, which is triggered by the LT's input x_2, outputs $o_{2,1}$ to FSM_2, which responds with input $a_{2,1}$ triggering $e4$. When $e4$ is traversed, it outputs $o_{1,2}$ to FSM_1, which responds with $a_{1,2}$ triggering $e8$. Transitions $e5$, $e6$, $e7$, $e9$, and $e10$, can be triggered directly by the LT. Transitions $e6$, $e7$, $e9$, and $e10$, generate outputs only to the LT, not to the semicontrollable interfaces. $e5$ generates output $o_{2,2}$ to FSM_2, which does not send any input back to the IUT (which is observationally equivalent to sending a null input to the IUT). For this example, we have:*

- $|V| = 3$, *which are* A, B, *and* C; $F = 2$, *which are* I_1 *and* I_2; $c_1 = 2, c_2 = 1$
- $T_{1,1} = \{e3\}, T_{1,2} = \{e8\}, T_{2,1} = \{e4\}, T_1 = T_{1,1} \cup T_{1,2} = \{e3, e8\}, T_2 = T_{2,1} = \{e4\}$
- $U_{1,1} = \{e1\}, U_{1,2} = \{e4\}, U_{2,1} = \{e2\}, U_1 = U_{1,1} \cup U_{1,2} = \{e1, e3\}, U_2 = U_{2,1} = \{e2\}$
- $W_{2,2} = \{e5\}; A_1 = \{a_{1,1}, a_{1,2}\}, A_2 = \{a_{2,1}\}; O_1 = \{o_{1,1}, o_{1,2}\}, O_2 = \{o_{2,1}, o_{2,2}\}$

4.2 Impact of buffer sizes on feasibility

Let us first assume that there is a separate FIFO buffer in a semicontrollable interface I_i. During testing, a buffer may be empty or store an arbitrary sequence of inputs to the IUT generated indirectly through the i-th semicontrollable interface. Then the entire system can be modeled by G (which represents the IUT's FSM) and the variables $\omega_1, \omega_2, \ldots, \omega_F$ representing the test context. Each ω_i has a distinct value for each permutation of inputs that the i-th buffer can hold.

If the buffer sizes at the F semicontrollable interfaces are infinite, each variable ω_i can have an infinite number of values. In this case, even the reachability analysis (deciding whether a given state is reachable from the initial state), which is an easier problem than finding a minimum cost traversal of G, becomes undecidable [12]. If the buffer sizes are finite, in which case $\omega_1, \omega_2, \ldots, \omega_F$ have finite domains, the reachability analysis is PSPACE-complete for the most general case [12].

Given the difficulty of analyzing G and F variables, let us explore the possibility of modeling the system as an FSM, represented by $G'(V', E')$ with the maximum number of nodes $|V'|_{max}$ equal to $|V| * \prod_{i=1}^{F} B(i)$, where $B(i)$ is the maximum possible number of states of the i-th buffer defined as $B(i) = (c_i^{b_i+1} - 1)/(c_i - 1)$ for $c_i > 1$, and $B(i) = 1 + b_i$ for $c_i = 1$. In general, if each $c_i = c > 1$, and each $b_i = b$, then

$$|V'|_{max} \quad = \quad |V| * O(c^{bF}) \tag{1}$$

Each vertex in V' is a tuple consisting of an original vertex in V and a set of values of variables $\omega_1, \omega_2, \ldots, \omega_F$ (this set is called a *configuration*). As indicated by (1), the maximum number of nodes in G' grows exponentially with the number of semicontrollable interfaces F and the buffer size b. Clearly, the conversion from G to G' is not feasible for the general case. However, for a constrained environment (Section 5), G' can be constructed efficiently, and test generation techniques can be applied to it.

5 Objectives for practical test system

This section shows through the detailed examples how a test sequence derived from the model of Section 4 may become nondeterministic. To avoid nondeterminism in test sequences, two practical objectives for a practical test system are introduced in Section 5.1. Diagnostic issues are discussed in Section 5.2.

5.1 Buffering inputs at semicontrollable interfaces during testing

The model of Section 4 utilizes FIFO-type buffers in semicontrollable interfaces. In practice, in addition to (or instead of) FIFO buffers, semicontrollable interfaces may include interrupt-driven mechanisms with multiple buffers in one interface. This freedom in implementing interfaces is mostly due to the interface not being part of a protocol specification. Therefore, test sequences generated for an IUT with only FIFO-type buffers become nondeterministic for IUTs using different interface implementations.

Example (cont'd): *Consider the following potential test sequence for the IUT of Figure 3:*

$$e7, e2, \underline{e4, e1, e8, e7}, e5, e3, e9, e10, e6 \tag{2}$$

As can be verified by Table 1, when the underlined portion of the above test sequence traverses $e4$*, input* $a_{1,2}$ *will be buffered at* I_1*. Subsequently, when* $e1$ *is executed with output* $o_{1,1}$ *to* I_1*, the buffer at* I_1 *should contain* $[a_{1,2}, a_{1,1}]$ *(i.e.,* $a_{1,2}$ *in front of the buffer). The IUT is expected to be in state* B *with* $e8$ *to be tested next. This sequence is only realizable under the assumption that inputs* $a_{1,2}$ *and* $a_{1,1}$ *are stored at* I_1 *in the FIFO order, i.e.,*

$[a_{1,2}, a_{1,1}]$. *In practice, however, this may not be the case for all implementations. Since the response time of a semicontrollable interface is unknown, it is possible that, after traversing $e4$ and $e1$, the buffer at I_1 will contain $[a_{1,1}, a_{1,2}]$. Then, after $e1$ is traversed, $e8$ cannot be triggered by $a_{1,2}$, because $a_{1,1}$ improperly blocks $a_{1,2}$ from being available. Transition $e8$ will cause the IUT to fail because the model assumes one type of buffering, and the implementation uses a different type. Clearly, the test sequence (2) is not realizable without FIFO-type buffers.*

This nondeterminism in a test sequence due to multiple inputs at one interface can be eliminated if a test sequence never creates a situation where more than one input will be stored in a given buffer (i.e., $b_i = 1$). Although each buffer's the capacity may be arbitrarily large, the technique presented in this paper preferably generates a test sequence with the objective Obj_1 : *at any given time, each semicontrollable buffer will store at most one input.* In this case, the maximum number of nodes in G' becomes $|V'|_{max} = |V| * \prod_{i=1}^{F}(c_i + 1)$. If Obj_1 cannot be achieved due to heavy interactions among the FSM_i (and possibly with uncontrollable inputs), the number of stored inputs will not be limited to one by a test sequence, and therefore, the test sequence will most likely have nondeterminism.

In a practical testing environment, F, the number of semicontrollable interfaces, is expected to be small. For most cases, the (N)-layer IUT interacts only with an (N+1)-layer implementation and several semicontrollable timers. Typically, for each timer, the only output is the timeout, which defines $c_i = 1$. Therefore, for small F and c_i, the size of G' is only a small multiplicant of G.

Let us now consider the number of inputs that can be buffered simultaneously at all of an IUT's semicontrollable interfaces. A test sequence may cause several inputs being buffered at the same time at several semicontrollable interfaces. If Obj_1 is satisfied, each buffer stores at most one input during testing, and the nondeterminism due to multiple inputs stored in a buffer is avoided. However, nondeterministic behavior of the system during testing may still occur because of the IUT's interactions with multiple interfaces (each interface with at most one stored input).

Example (cont'd): *Consider a potential test sequence (3) for the IUT of Figure 3. The test sequence was generated for an IUT with buffer sizes of 1 at I_1 and I_2:*

$$\underline{e1, e2, e3, e4}, e7, e8, e7, e5, e9, e10, e6 \tag{3}$$

The test sequence of (3) avoids the nondeterminism due to multiple stored inputs in a given buffer shown previously in (2), since it stores at most one input at a given interface. However, the test sequence of (3) may still be nondeterministic since the IUT is interacting with multiple interfaces simultaneously. Consider the underlined portion of (3). After $e1$ is traversed, input $a_{1,1}$ is buffered at I_1. Traversing $e2$ results in $a_{2,1}$ being buffered at I_2. Since $a_{1,1}$ was generated earlier than $a_{2,1}$, the test sequence expects transition $e3$ is expected to be triggered before $e4$. In reality, due to the unknown response time of the interfaces, $a_{2,1}$ may be applied to the IUT earlier than, later than, or simultaneously with $a_{1,1}$. In this case, the behavior of the overall system becomes nondeterministic under a test sequence, thereby making the test sequence unrealizable.

To avoid this type of nondeterministic behavior of the IUT during testing, the model presented in Section 4 will be used to generate tests with a second objective Obj_2: *at any time, the test sequence will cause only a single input to be stored in only one of the IUT's semicontrollable interfaces.* In other words, when Obj_2 can be achieved, only a single message is in transit when a test sequence is applied. In this case, the maximum number of nodes in V' is $|V'|_{max} = |V| * (1 + \sum_{i=1}^{F} c_i)$.

The objectives of Obj_1 and Obj_2 address the types of nondeterminism that will occur when an IUT moves into a state where there are multiple inputs stored in multiple interfaces. A valid test sequence which satisfies Obj_1 and Obj_2 will not bring the IUT into a state with multiple inputs pending. Instead, a valid test sequence should traverse the IUT transitions in such an order that avoids these situations.

This type of nondeterminism caused by multiple inputs is one of several types of nondeterministic behavior that can occur in a system with multiple communicating modules. The discussion of various types of nondeterministic behavior, algorithms to detect them, and guidelines how to rewrite specifications to avoid nondeterminism are discussed in [16] in the context of Formal Description Technique Estelle [4].

It is important to note that the minimum-length test sequences satisfying Obj_1 and Obj_2 (i.e., at most one input at only one semicontrollable interface utilized at any time) will likely be longer than the minimum-length test sequences for the unrestricted case. However, tests satisfying these two objectives can be used for testing implementations regardless of their interface structure, avoiding nondeterministic behavior of the SUT during testing. Test generation is further discussed in Section 6.

5.2 Diagnostic issues during testing

As presented in Section 2, during testing an IUT may interact with several semicontrollable interfaces. Testing is performed under the assumption that all FSM implementations other than the IUT conform to their specifications. Otherwise, it is difficult to tell whether failure occurs in the IUT, or in the external FSM implementation, or at the semicontrollable interface between them.

Example (cont'd): *Consider the beginning of the test sequence (2) for the IUT of Figure 3:*

$$e7, e2, e4, e1, ...$$

When this part of the test sequence is applied to the IUT, traversal of $e2$ should cause FSM_2 to send back input $a_{2,1}$. The IUT will move to state C with $a_{2,1}$ buffered at I_2. Suppose that a faulty implementation incorrectly contains $a_{1,1}$ instead of $a_{2,1}$ at I_2. Then in state C transition $e3$ will be triggered by $a_{1,1}$, and the IUT will remain in C instead of moving to A after $e4$'s traversal, even if $e2$, $e3$, and $e4$ are implemented correctly. The tester cannot distinguish whether $e2$'s, $e3$'s, or $e4$'s implementation is faulty, or FSM_2 is not conformant to its specification, or the semicontrollable interface I_2 malfunctioned. Although FSM_2 and I_2 can be assumed fault-free as part of the test context [35], a test verdict should be based on the test context's correctness only when necessary.

This practical concern for problem diagnosis suggests the following testing guideline: "Test as many transitions as possible without interactions at semicontrollable interfaces." Transitions preferably should be tested when there are no inputs buffered at the semicontrollable interfaces. As a result of this guideline, a minimum cost test sequence generation can be formulated (under Section 5's considerations) and solved as a Rural Chinese Postman Problem [17], as discussed in Section 6.

6 Minimum-cost test generation

This section discusses test generation for the practical testing environment as described in Section 5. A method to obtain a test sequence as a solution to the Rural Chinese Postman Problem [1, 17] on graph $G'(V', E')$ is shown; this method is then applied to an example practical testing environment.

The graph G' is built by the algorithm (referred to as *CONVERT-SEMI-INT* henceforth) presented and analyzed in [10, 11]. The algorithm creates a new state $v' \in V'$ from two components: the original state $v \in V$, and the current configuration of buffers modeled by variables $\omega_1, \omega_2, \ldots, \omega_F$. In this process, all possible buffer configurations with up to b_i inputs in buffer B_i at I_i are constructed by examining in a breadth-first-search manner all outgoing edges of v. One or more copies are created in E' for each edge $e \in E$, based on e's class. In general, each edge in E belongs to one of the four classes [11] defined based on the source and destination (FSM_i or/and LT) of the edge's input and output(s) as follows:

- *Class 1:* e is triggered by an input from and generates output(s) to an LT.

Step	*Edge name*	*Input from*	*Output to*	*Step*	*Edge name*	*Input from*	*Output to*
$\rightarrow 1$	e1.0	$LT?x_1$	$FSM_1!o_{1,1}$	8	e7.2	$LT?x_7$	$LT!y_7$
2	e5.1	$LT?x_5$	$FSM_2!o_{2,2}$	$\rightarrow 9$	e8.2	$FSM_1?a_{1,2}$	$LT!y_8$
$\rightarrow 3$	e3.1	$FSM_1?a_{1,1}$	$LT!y_3$	10	e7.0	$LT?x_7$	$LT!y_7$
$\rightarrow 4$	e6.0	$LT?x_6$	$LT!y_6$	$\rightarrow 11$	e5.0	$LT?x_5$	$FSM_2!o_{2,2}$
$\rightarrow 5$	e7.0	$LT?x_7$	$LT!y_7$	$\rightarrow 12$	e9.0	$LT?x_9$	$LT!y_9$
$\rightarrow 6$	e2.0	$LT?x_2$	$FSM_2!o_{2,1}$	$\rightarrow 13$	e10.0	$LT?x_{10}$	$FSM_2!y_{10}$
$\rightarrow 7$	e4.3	$FSM_2?a_{2,1}$	$FSM_1!o_{1,2}$	14	e6.0	$LT?x_6$	$LT!y_6$

Table 2: Minimum-length test sequence for the IUT of Figure 3.

- *Class 2:* e is triggered by an input from an LT and generates an output $o_{q,l}$ (buffered in B_q to create a new configuration) at I_q.
- *Class 3:* e is triggered by $a_{p,k}$ (extracted from B_p to create a new configuration) from I_p and generates output(s) to an LT.
- *Class 4:* e is triggered by an input $a_{p,k}$ from I_p and generates an output $o_{q,l}$ at I_q.

The algorithm's running time is shown [10] to be $O(c * F * |E|)$ if the objectives of Obj_1 and Obj_2 can be satisfied, and $O(c^{bF} * |E|)$ for $c > 1$ otherwise.

Each path of G', which consists of edges in E', can be proven valid [10], i.e., for any of its composite edges $e' \in E'$ the following hold true: (1) e' has no race conditions, (2) if e' is triggered by a buffered input $a_{i,j}$, this input is consumed from the buffer, (3) the buffer that should store e''s output is not full in a given state, and (4) if e' is fireable by input $a_{i,j}$, this input is the first one buffered in the configuration corresponding to e''s start state. Therefore, G' can be shown [10] to be a *minimal valid* representation of the system defined by G and $\omega_1, \cdots, \omega_F$, which implies that each path of G' is a valid path of G, and that no invalid paths of G are included in G'. A test sequence obtained from G' does not contain any race conditions, as proven in [10] and illustrated through an example at the end of this section.

For graphs $G(V, E)$ and $G'(V', E')$, a test sequence is derived by obtaining the following goal: "find a minimum cost tour of G' in which each original edge from G included in G' is covered at least once." (Note that if an IUT's transition cannot be covered within a given test context, the algorithm for graph conversion will not include this transition in G'.) Let E'_c be the set of edges defined based on the practical considerations (Section 5.2) as containing copies incident to nodes corresponding to configurations with empty buffers (where possible). It is clear that E'_c will include at least one copy of each edge in E. Therefore, obtaining the above goal is equivalent to finding a minimum cost tour of G' that includes each transition in E'_c, the set of *mandatory* edges, at least once, and each transition in $(E' - E'_c)$, the set of *optional* edges, zero or more times (so called Rural Chinese Postman Problem (RCPP) [17], with an efficient solution presented by Aho et al. [1]). It can be shown that E'_c as defined above is a weakly-connected subset of E'; therefore, a polynomial-time solution to RCPP formulated on G' and E'_c exists.

Example (cont'd): *Consider the graph of Figure 3. After conversion to G' (Figure 4), each state is replaced with at most four copies—each corresponding to the buffer configuration at a semicontrollable interface. Each edge e is annotated as $e.x$, where $x = 0, 1, 2, 3$, depending on the input buffered in the $e.x$'s start state, as shown in Figure 4. Given graphs G and G', the sets E and E' are as follows:*

$$E = \{e1, e2, e3, e4, e5, e6, e7, e8, e9, e10\} \quad (4)$$
$$E' = \{e1.0, e2.0, e3.1, e4.3, e5.0, e5.1, e6.0, e7.0, e7.2, e8.2, e9.0, e10.0, e10.1\} \quad (5)$$

To build the set of mandatory edges to be included in a test sequence, we adopt the approach discussed in Section 5.2. In G', edges $e5$ and $e7$ appear multiple times. The solid edges in Figure 4 are the mandatory edges that are incident to nodes that correspond to the case where both buffers are empty, i.e., $e5.0$ and $e7.0$. The copies that

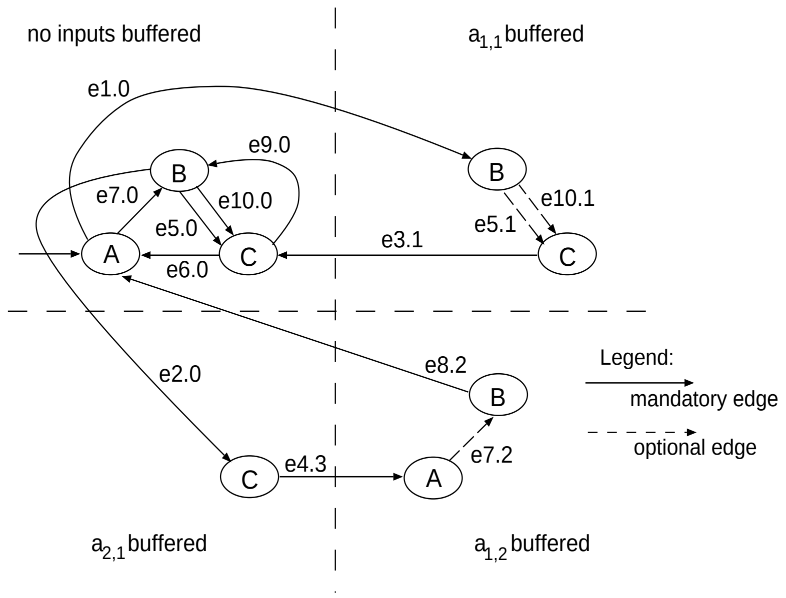

Figure 4: Graph transformation applied to the graph of Figure 3. Mandatory and optional edges appear in solid and dashed lines, respectively.

can be traversed only when either buffer contains an input are shown in dashed line: $e5.1$ *and* $e7.2$*. These are the optional edges, which will be included in the test sequence only when necessary. In this example we have:*

$$E'_c \quad = \quad \{e1.0, e2.0, e3.1, e4.3, e5.0, e6.0, e7.0, e8.2, e10.0\} \tag{6}$$

Given sets E' *and* E'_c *defined by (5) and (6), the Aho et al. optimization technique gives the minimum length test sequence for* G' *shown in Table 2. Steps with* $(\rightarrow)$ *indicate that an edge is tested in this step. Note that, for simplicity, the UIO sequences [20] for state verification are not included in this sequence.*

7 Controlling FSMs associated with semicontrollable interfaces

The analysis presented thus far is focused on controlling an IUT where a semicontrollable FSM_i interacts with the IUT. In this system, FSM_i generates a desired input to the IUT as a response to the IUT's stimulus without a tester's explicit control.

This section enhances the model presented in Section 4 by including preambles and postambles to control both the IUT and FSM_i: bringing FSM_i into a desired state by a preamble (Section 7.1), and bringing FSM_i from a given state into its initial state by a postamble (Section 7.2). The graph augmentation with preambles and postambles needed to obtain a near-minimum cost test sequence is proposed in Section 7.3. Section 7.4 discusses the validity of paths in the augmented graph. Finally, test coverage and cost effectiveness of the proposed methodology is discussed in Section 7.5.

Note that this augmentation does *not* aim to test the the FSMs that are communicating with the IUT; the purpose of the augmentation is to move a semicontrollable FSM into a state where a matching transition (defined below) can be utilized to test the IUT.

Let $\bar{G}_i(\bar{V}_i, \bar{E}_i)$ be the graph representing the semicontrollable interface I_i with FSM_i. Let $\bar{U}_{i,j}$ be the set of edges in $\bar{E}_i$ that are triggered by input $o_{i,j}$ from the IUT, and that generate output $a_{i,j}$ to the IUT.

Consider a transition $u_{i,j} = (u^1_{i,j}, u^2_{i,j}) \in U_{i,j}$ in the IUT. The methodology presented in the paper up to this point is applicable to a testing system where given $u_{i,j}$, there is always a matching transition $\bar{p}_{i,j} = (\bar{p}^1_{i,j}, \bar{p}^2_{i,j}) \in \bar{U}_{i,j}$ in FSM_i. This constraint implies that upon applying input $o_{i,j}$ to FSM_i, there is always an output $a_{i,j}$ from FSM_i. A trivial case where this holds true is an FSM_i with each state having an edge $\bar{p}_{i,j}$. In practice, however, FSM_i may be in a state where $\bar{p}_{i,j}$ is undefined, making the controllability over I_i more difficult.

One possible solution is to model an IUT combined with all semicontrollable interfaces as a single FSM. This solution is infeasible, since it amounts to multiplying the IUT's FSM and $FSM_1, \ldots, FSM_F$, resulting in a state explosion problem. Fortunately, it is possible to achieve controllability in such a system by restricting the state space in FSM_i that is reachable by the IUT. The goal of testing is to test each $t_{i,j} = (t^1_{i,j}, t^2_{i,j}) \in T_{i,j}$ in the IUT. For a given $t_{i,j}$, we suggest the following test steps:

- the IUT is in any state $v \in V$, and FSM_i in its initial state $\bar{v}_0$
- for a given $u_{i,j}$, move the IUT from state v to state $u^1_{i,j}$, and FSM_i from state $\bar{v}_0$ to state $\bar{p}^1_{i,j}$
- trigger $u_{i,j}$ by applying its input from the lower tester. $u_{i,j}$ generates $o_{i,j}$ to FSM_i; transition $\bar{p}_{i,j}$ consumes $o_{i,j}$ and outputs $a_{i,j}$ to the IUT. Now the IUT is in state $u^2_{i,j}$ and FSM_i in state $\bar{p}^2_{i,j}$
- move the IUT to state $t^1_{i,j}$ (FSM_i remains in state $\bar{p}^2_{i,j}$)
- buffered input $a_{i,j}$ triggers $t_{i,j}$, and the IUT moves to state $t^2_{i,j}$
- move the IUT from $t^2_{i,j}$ to state $x \in V$, and move FSM_i from $\bar{p}^2_{i,j}$ to its initial state $\bar{v}_0$

The above sequence suggests that the tester keep FSM_i in its initial state $\bar{v}_0$ while the IUT is being tested; only when $u_{i,j}$ is to be traversed, does the tester move the FSM_i to state $\bar{p}^1_{i,j}$. After $\bar{p}_{i,j}$ is traversed, the FSM_i is brought back to $\bar{v}_0$. This restriction will be referred to as the *controllability restriction*.

7.1 Bringing semicontrollable FSM to a desired state

Sabnani et al. [30] introduce an algorithmic procedure for checking the safety properties of communication protocols. The procedure takes a collection of communicating FSMs as input, and produces a composite output FSM by doing incremental composition and reduction. The state space of the composite FSM, which is observationally equivalent to the input FSMs, is reduced by several orders of magnitude.

Given $\bar{G}_i$ for FSM_i, let $G^*_i(V^*_i, E^*_i)$ be the graph obtained by combining G for the IUT and $\bar{G}_i$ as given by Sabnani et al. [30]. Let P_{IUT}, $\bar{P}_i$, and P^*_i, be the sets of paths of graphs G, $\bar{G}_i$, and G^*_i, respectively. Given $G^*_i(V^*_i, E^*_i)$ and the initial vertex $\bar{v}_0 \in \bar{V}_i$, let $pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j}) \in P^*_i$, where $v \in V$, $u_{i,j} = (u^1_{i,j}, u^2_{1,j}) \in U_{i,j} \cup W_{i,j}$, and $\bar{p}_{i,j} = (\bar{p}^1_{i,j}, \bar{p}^2_{i,j}) \in \bar{U}_{i,j}$, be a shortest path originating in $(v, \bar{v}_0) \in V^*_i$ such that

$$(v, \bar{v}_0) \overset{pr_{i,j}}{\rightsquigarrow} (u^1_{i,j}, \bar{p}^1_{i,j}) \tag{7}$$

$pr_{i,j}$ will be called a *preamble* of state $u^1_{i,j}$. For example, Figure 5 illustrates $pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j})$ and $pr_{i,j}(v, u^1_{i,j}, \bar{q}^1_{i,j})$, which are two preambles defined for state $u^1_{i,j}$ in the IUT. The preamble $pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j})$ consists of a path from v to $u^1_{i,j}$ in the IUT, and the corresponding path from $\bar{v}_0$ to $\bar{p}^1_{i,j}$ in the FSM_i. Similarly, the preamble $pr_{i,j}(y, u^1_{i,j}, \bar{q}^1_{i,j})$ is a combination of a path from y to $u^1_{i,j}$ in the IUT, and the corresponding path from $\bar{v}_0$ to $\bar{q}^1_{i,j}$ in the FSM_i. All possible such preambles must be considered to minimize the total test sequence cost, as described later in this section.

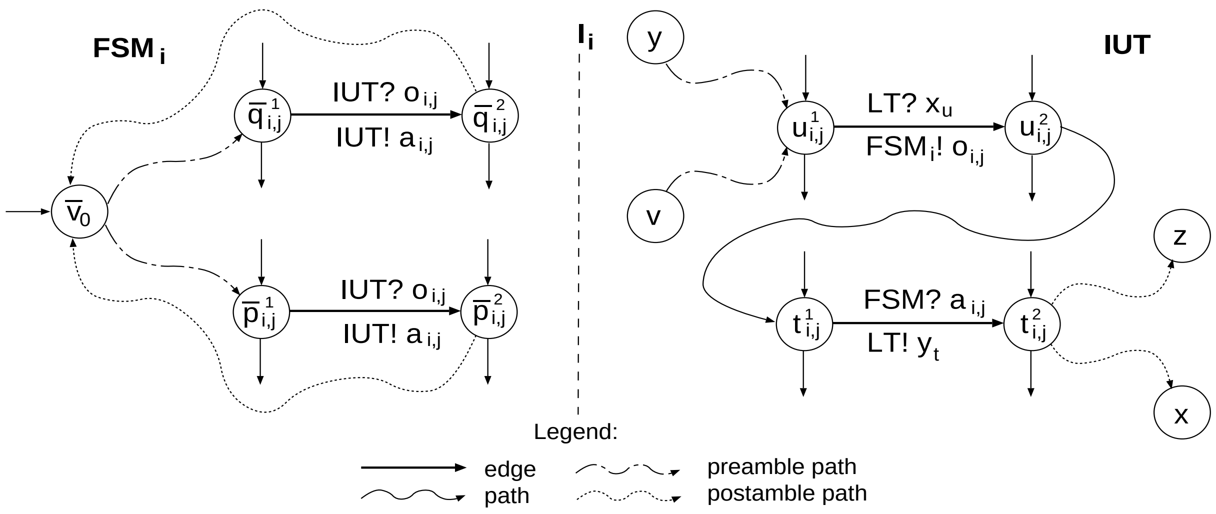

Figure 5: Two preambles ($pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j})$ and $pr_{i,j}(v, u^1_{i,j}, \bar{q}^1_{i,j})$) for $u^1_{i,j}$, and two postambles ($po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j})$ and $po_{i,j}(t^2_{i,j}, \bar{q}^2_{i,j})$) for $t^2_{i,j}$.

Let $P_{i,j}$, the set of all preambles associated with $U_{i,j}$ and $W_{i,j}$, be defined as follows:

$$P_{i,j} \;=\; \{pr_{i,j} : (\exists v \in V)(\exists u_{i,j} \in U_{i,j} \cup W_{i,j})(\exists \bar{p}_{i,j} \in \bar{U}_{i,j})\; pr_{i,j} = pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j})\} \tag{8}$$

It is clear that $|P_{i,j}| = O(|V| * |U_{i,j} \cup W_{i,j}| * |\bar{U}_{i,j}|)$. To find the set of shortest paths for all $(v, \bar{v}_0) \in V_i^*$ to other nodes in $\bar{G}_i$ takes $O(|V|(|V_i^*| \lg |V_i^*| + |E_i^*|))$ time [6].

Recall from Section 5 that at any time a single input may be buffered in only one of the IUT's semicontrollable interfaces. The algorithm presented in [30] ensures that in state $u^1_{i,j}$ (and in state $\bar{p}^1_{i,j}$ in FSM_i) no inputs are buffered at I_i. This implies that after traversing any $pr(v, u^1_{i,j}, \bar{p}^1_{i,j})$ followed by $u_{i,j} \in U_{i,j}$, only input $a_{i,j}$ is buffered at I_i (or no input if $u_{i,j} \in W_{i,j}$). The IUT is in state $u^2_{i,j}$ and FSM_i in state $\bar{p}^2_{i,j}$. Therefore, traversing a preamble as defined in (7) will not generate any extra inputs buffered at I_i, which will enable the application of the algorithm *CONVERT-SEMI-INT*.

After traversing $pr(v, u^1_{i,j}, \bar{p}^1_{i,j})$ followed by $u_{i,j} \in U_{i,j}$, FSM_i remains in state $\bar{p}^2_{i,j}$ until $a_{i,j}$ is consumed at I_i by a transition $t_{i,j} = (t^1_{i,j}, t^2_{i,j}) \in T_{i,j}$. After $t_{i,j}$ is triggered, FSM_i must be brought back to state $\bar{v}_0$. The postamble to bring FSM_i back to its initial state is discussed next.

7.2 Bringing semicontrollable FSM back to initial state

Given $G_i^*(V_i^*, E_i^*)$ and the collection of sets $T_{i,j}$, let $po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j}) \in P_i^*$, where $t_{i,j} = (t^1_{i,j}, t^2_{i,j}) \in T_{i,j}$ and $\bar{p}_{i,j} = (\bar{p}^1_{i,j}, \bar{p}^2_{i,j}) \in \bar{U}_{i,j}$, be a shortest path of G_i^* such that

$$(t^2_{i,j}, \bar{p}^2_{i,j}) \overset{po_{i,j}}{\leadsto} (v, \bar{v}_0),\; v \in V \tag{9}$$

$po_{i,j}$ will be called a *postamble* of an edge state $t^2_{i,j}$. For example, Figure 5 depicts two postambles defined for state $t^2_{i,j}$ in the IUT. The postamble $po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j})$ consists of a path from $t^2_{i,j}$ to x in the IUT, and the corresponding path from $\bar{p}^2_{i,j}$ to $\bar{v}_0$ in FSM_i. Similarly, the postamble $po_{i,j}(t^2_{i,j}, \bar{q}^2_{i,j})$ is a combination of a path from $t^2_{i,j}$ to z in the IUT, and the corresponding path from $\bar{q}^2_{i,j}$ to $\bar{v}_0$ in FSM_i.

For a given $t_{i,j}$, there may be multiple states in FSM_i in which $o_{i,j}$ can be consumed and $a_{i,j}$ output to the IUT. Depending on which preamble is used to bring the IUT into $u^1_{i,j}$, a matching postamble must be utilized to bring FSM_i back to $\bar{v}_0$. For example, in Figure 5, the preamble $pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j})$ requires the postamble $po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j})$ to bring the FSM_i from $\bar{p}^2_{i,j}$ to $\bar{v}_0$ (which causes the IUT to end up in state x). The other postamble shown in Figure 5, $po_{i,j}(t^2_{i,j}, \bar{q}^2_{i,j})$, cannot be used in this case. Similarly, the preamble $pr_{i,j}(v, u^1_{i,j}, \bar{q}^1_{i,j})$ requires the postamble $po_{i,j}(t^2_{i,j}, \bar{q}^2_{i,j})$ and prohibits the use of $po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j})$.

In the case of $w_{i,j} = (w^1_{i,j}, w^2_{i,j}) \in W_{i,j}$, the postamble defined by (9) starts from the state $w^2_{i,j}$, i.e.,

$$(w^2_{i,j}, \bar{p}^2_{i,j}) \overset{po_{i,j}}{\leadsto} (v, \bar{v}_0),\ v \in V \tag{10}$$

Let $Q_{i,j}$, the set of all postambles associated with $T_{i,j}$ and $W_{i,j}$, be defined as follows:

$$Q_{i,j} \quad = \quad \{po_{i,j} : (\exists t_{i,j} \in T_{i,j} \cup W_{i,j})(\exists \bar{p}_{i,j} \in \bar{U}_{i,j})\ po_{i,j} = po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j})\} \tag{11}$$

In the above, $|Q_{i,j}| = O(|T_{i,j} \cup W_{i,j}| * |\bar{U}_{i,j}|)$, which requires at most $O(|V|(|V^*_i| \lg |V^*_i| + |E^*_i|))$ time to compute [6].

7.3 Augmenting graph with preambles and postambles

To obtain a near-minimum cost test sequence, all possible preambles and their matching postambles must be considered, i.e., an augmentation must be defined to include all the preambles $pr_{i,j}$ and all the postambles $po_{i,j}$, for a given $\bar{p}_{i,j} = (\bar{p}^1_{i,j}, \bar{p}^2_{i,j}) \in \bar{U}_{i,j}$ in FSM_i. Let $\bar{p}_{i,j}$ be chosen by minimizing an objective function $f_{i,j} : \bar{U}_{i,j} \to \mathcal{R}^+$ (where $\mathcal{R}^+$ is the set of non-negative real numbers), which computes the average length of a preamble ending at $\bar{p}^1_{i,j}$, and a postamble starting at $\bar{p}^2_{i,j}$ in FSM_i.:

$$f_{i,j}(\bar{p}_{i,j}) = 0.5 * (\frac{\sum_{v \in V} \sum_{u_{i,j} \in U_{i,j} \cup W_{i,j}} |pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j})|}{|V| * (|U_{i,j}| + |W_{i,j}|)} + \frac{\sum_{t_{i,j} \in T_{i,j} \cup W_{i,j}} |po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j})|}{|T_{i,j}| + |W_{i,j}|}) \tag{12}$$

Let us augment graph G with preambles (prior to conversion to G' by the algorithm *CONVERT-SEMI-INT*) for the edges in $U_{i,j}$ and $W_{i,j}$ as follows:

1. $\forall u_{i,j} \in U_{i,j} \cup W_{i,j}$ split $u^1_{i,j}$ into $u^{1,I}_{i,j}$ and $u^{1,II}_{i,j}$
2. $\forall pr_{i,j}(v, u^1_{i,j}, \bar{p}^1_{i,j}) \in P_{i,j}$ create an edge $(v, u^{1,II}_{i,j})$
3. replace $u_{i,j} = (u^1_{i,j}, u^2_{i,j})$ with $u_{i,j} = (u^{1,II}_{i,j}, u^2_{i,j})$
4. all incoming (outgoing) edges of $u^1_{i,j}$ become incoming (outgoing) edges of $u^{1,I}_{i,j}$

Using postambles, we continue the augmentation of G for the edges in $T_{i,j}$ as follows (to augment G with postambles for the edges in $W_{i,j}$, replace $t_{i,j}$ with $w_{i,j}$):

1. $\forall t_{i,j} \in T_{i,j}$ split $t^2_{i,j}$ into $t^{2,I}_{i,j}$ and $t^{2,II}_{i,j}$
2. $\forall po_{i,j}(t^2_{i,j}, \bar{p}^2_{i,j}) \in Q_{i,j}$ create an edge $(t^{2,II}_{i,j}, v)$
3. replace $t_{i,j} = (t^1_{i,j}, t^2_{i,j})$ with $t_{i,j} = (t^1_{i,j}, t^{2,II}_{i,j})$
4. all incoming (outgoing) edges of $t^2_{i,j}$ become incoming (outgoing) edges of $t^{2,I}_{i,j}$

As mentioned earlier, the running time of the combined algorithms for finding sets $P_{i,j}$ and $Q_{i,j}$ for all transition classes in all interfaces is given by [6] as $O(\sum_{i=1}^{F}(|V|(|V^*_i| \lg |V^*_i| + |E^*_i|))$.

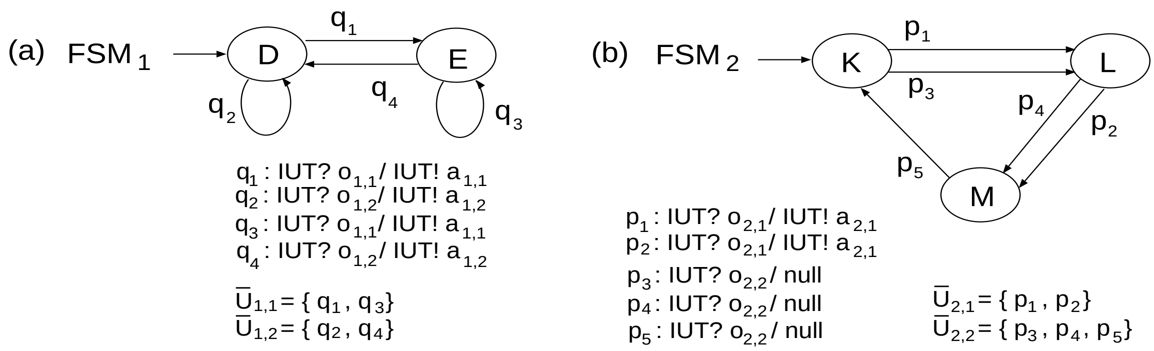

Figure 6: The FSM for the FSM_1 (a) and the FSM_2 (b) in the system of Figure 3.

preamble	*start state*	*end state*	*IUT edges*	FSM_2 *edges*
$pr_{2,1}(A,B,K), pr_{2,2}(A,B,K)$	(A,K)	(B,K)	$e7$	$\emptyset$
$pr_{2,1}(B,B,K), pr_{2,2}(B,B,K)$	(B,K)	(B,K)	$\emptyset$	$\emptyset$
$pr_{2,1}(C,B,K), pr_{2,2}(C,B,K)$	(C,K)	(B,K)	$e6, e7$	$\emptyset$
$pr_{2,1}(A,B,L), pr_{2,2}(A,B,L)$	(A,K)	(B,L)	$e7, e5, e9$	p_3
$pr_{2,1}(B,B,L), pr_{2,2}(B,B,L)$	(B,K)	(B,L)	$e5, e9$	p_3
$pr_{2,1}(C,B,L), pr_{2,2}(C,B,L)$	(C,K)	(B,L)	$e9, e5, e9$	p_3
$pr_{2,2}(A,B,M)$	(A,K)	(B,M)	$e7, e5, e9, e5, e9$	p_3, p_4
$pr_{2,2}(B,B,M)$	(B,K)	(B,M)	$e5, e9, e5, e9$	p_3, p_4
$pr_{2,2}(C,B,M)$	(C,K)	(B,M)	$e9, e5, e9, e5, e9$	p_3, p_4
postamble	*start state*	*end state*	*IUT edges*	FSM_i *edges*
$po_{2,1}(A,L)$	(A,L)	(C,K)	$e7, e8, e7, e5, e9, e5$	p_4, p_5
$po_{2,1}(A,M)$	(A,M)	(C,K)	$e7, e8, e7, e5$	p_5
$po_{2,2}(C,L)$	(C,L)	(C,K)	$e9, e5, e9, e5$	p_4, p_5
$po_{2,2}(C,M)$	(C,M)	(C,K)	$e9, e5$	p_5
$po_{2,2}(C,K)$	(C,K)	(C,K)	$\emptyset$	$\emptyset$

Table 3: Preambles and postambles for the FSMs of Figure 6.

7.4 Checking validity of preambles and postambles

As proven in [10], all paths in graph G' are valid. Augmenting graph G with preambles and postambles results in adding a new class of edges to G—the concatenated edges consisting of preambles' and postambles' composite edges. The algorithm *CONVERT-SEMI-INT* should be modified accordingly to account for the existence of concatenated edges. In particular, to avoid introducing invalid paths into G', it must be verified that each concatenated edge to be included in G' is a valid path.

All preambles associated with $U_{i,j}$ are necessarily valid paths in G'. A test sequence satisfying the objectives of Obj_1 and Obj_2 requires that, at any given time, only one input can be buffered at I_i. Since each preamble associated with $U_{i,j}$ is followed in G' by an edge in $U_{i,j}$, whose traversal causes $a_{i,j}$ to be buffered at I_i, the algorithm makes sure that each such preamble starts and ends in states corresponding to the configurations with empty buffers. Therefore, there is no buffered input that can disrupt the preamble's traversal. Similarly, each postamble associated with $T_{i,j}$ is always valid in G'. However, both preambles and postambles associated with $W_{i,j}$ may or may not be valid in G' and should be checked for validity by the algorithm *CONVERT-SEMI-INT*.

Example (cont'd): *Consider the testing environment depicted in Figure 3. The FSM_1 and FSM_2 are shown in Figure 6. Suppose that the IUT's transition $e5$, which outputs $o_{2,2}$ to FSM_2, triggers FSM_2's transitions p_3, p_4, or p_5. Among the FSM_2's transitions, the ones in $\bar{U}_{2,1}$ generate outputs to the IUT. For the IUT's edges in $U_{1,1}$ and*

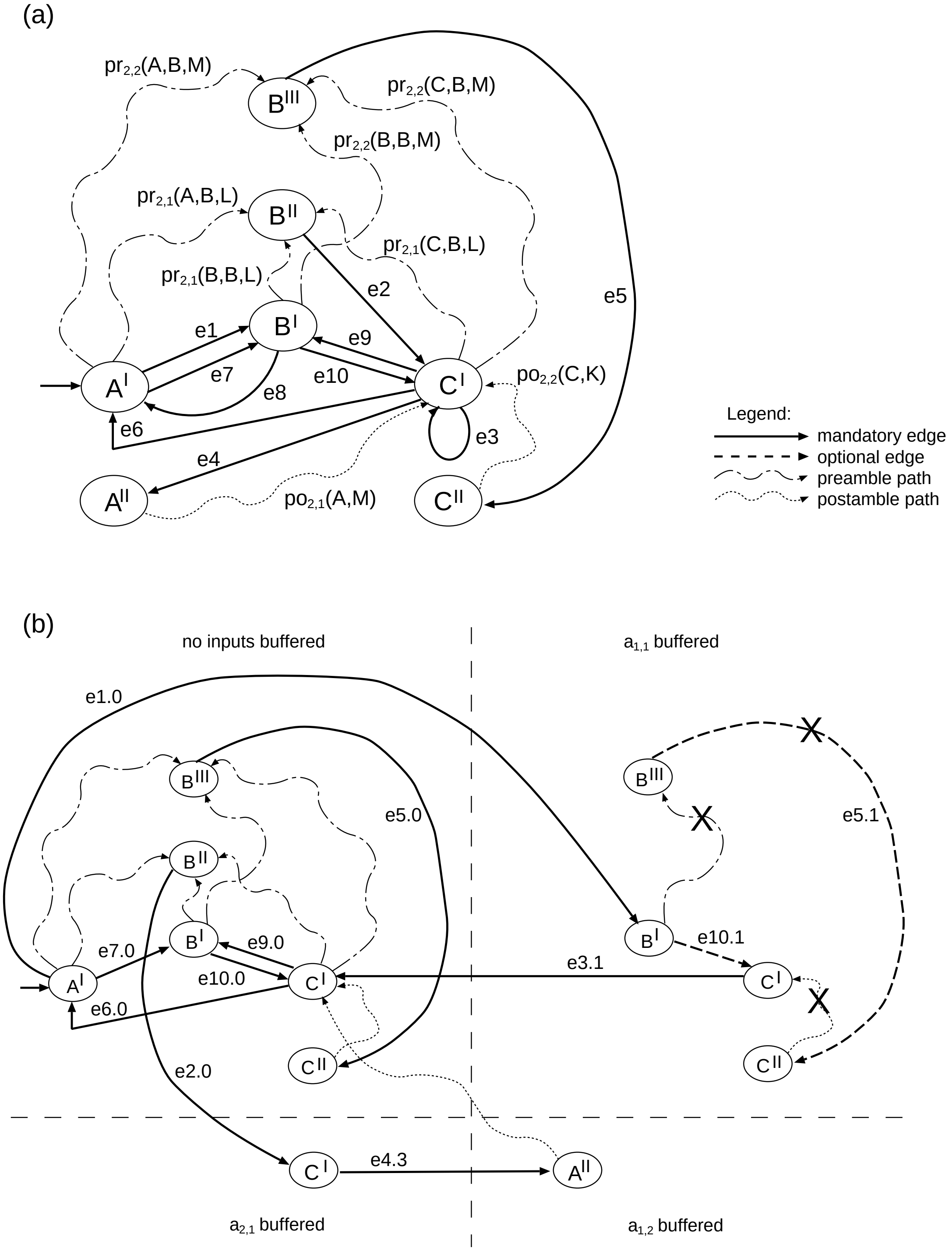

Figure 7: (a) Augmenting graph G with preambles and postambles; (b) G' obtained from the augmented G by using the algorithm *CONVERT-SEMI-INT*.

$U_{1,2}$ there is always a matching transition in the FSM_1. However, for the IUT's edges in $U_{2,1}$, the two matching transitions are defined only for states K and L. Therefore, utilizing transitions in $\bar{U}_{2,1}$ and $\bar{U}_{2,2}$ requires the use of preambles and postambles. Table 3 shows the preambles and postambles for the testing system of Figure 3. For this system, $U_{2,1} = \{e2\}, T_{2,1} = \{e4\}, W_{2,2} = \{e5\}, \bar{U}_{2,1} = \{p_1, p_2\}$, and $\bar{U}_{2,2} = \{p_3, p_4, p_5\}$.

To augment graph G, we first compute the value of the objective function $f_{2,1}$ (defined by (12)) for p_1 and p_2: $f_{2,1}(p_1) = 4.5$ and $f_{2,1}(p_2) = 4.33$. Since p_2 produces slightly shorter preambles and postambles, it will be used for augmentation of G. Similarly, we compute $f_{2,2}$ for p_3, p_4, and p_5: $f_{2,2}(p_3) = 3.5$, $f_{2,2}(p_4) = 3.33$, and $f_{2,2}(p_5) = 3.33$. Let us pick p_5 to augment G. By applying the technique presented in Section 7.3, we first augment G with preambles and postambles (Figure 7 (a)), and then convert G to G' (Figure 7 (b)) by the algorithm CONVERT-SEMI-INT. *The preamble $pr_{2,2}(B, B, M)$, which consists of edges $e5$, $e9$, $e5$, and $e9$, is a valid path with respect to the configuration with all buffers empty. However, it is an invalid path with respect to configurations with $a_{1,2}$ (or $a_{1,1}$) buffered at I_1, or $a_{2,1}$ buffered at I_2. For example, the preamble's edge $e9$ is invalid when $a_{1,1}$ is buffered at I_1, since in state C transition $e3$ will automatically trigger instead of $e9$ being triggered by a lower tester's input x_9. (Edges that would have been included in G' by the algorithm* CONVERT-SEMI-INT *had the preambles and postambles not been checked for validity, are shown in Figure 7 (b) as crossed with "X.")*

Finally, we find a rural Postman tour of G':

$$e1.0,\ e10.1,\ e3.1,\ e6.0,\ \underbrace{pr_{2,1}(A,B,L)}_{e7,e5,e9},\ e2.0,\ e4.3,\ \underbrace{po_{2,1}(A,M)}_{e7,e8,e7,e5},\ e9.0, \tag{13}$$

$$\underbrace{pr_{2,2}(B,B,M)}_{e5,e9,e5,e9},\ e5.0,\ \underbrace{po_{2,2}(C,K)}_{\emptyset},\ e6.0,\ e7.0,\ \underbrace{pr_{2,2}(B,B,M)}_{e5,e9,e5,e9},\ e5.0,\ \underbrace{po_{2,2}(C,K)}_{\emptyset},\ e6.0 \tag{14}$$

After replacing the preambles and the postambles with their composite edges, and dropping the suffixes of regular edges, the following near-minimum cost (Section 7.5) test sequence is obtained:

$$e1,\ e10,\ e3,\ e6,\ e7,\ e5,\ e9,\ e2,\ e4,\ e7,\ e8,\ e7,\ e5, e9,\ e5,\ e9,\ e5,\ e9,\ e5,\ e6,\ e7,\ e5,\ e9,\ e5,\ e9,\ e5,\ e6$$

7.5 Test coverage and cost effectiveness

In general, without limiting the number of possible state transitions in FSM_i during testing, finding a cost-effective test sequence for the IUT becomes infeasible: the number of potential test scenarios involving the IUT and all FSM_i's is prohibitively large. Even when the objective of Obj_2 (defined in Section 5.1) can be met, deriving a tour based on a part of a composite machine satisfying a requirement of a single message in transit would involve the testing of the entire state space of the semicontrollable FSMs. To avoid this inefficiency in test sequences, the so-called *controllability restriction* defined at the beginning of Section 7 suggests that FSM_i must be kept in its initial state and brought into a desired state only when needed, yielding test sequences of feasible length. The proposed approach is particularly efficient in the case where the state space of the semicontrollable FSMs is large, since there may be many transitions of the text context that are not needed to test the desired transitions of the IUT.

However, it may be argued that the controllability restriction does not make all IUT transitions testable. In particular, to bring FSM_i into a desired state may involve edges in $\bar{G}_i$ that require interactions between FSM_i and interfaces other than the IUT. Based on the controllability restriction on the FSM_i's state space, such edges will not be traversed during testing. If avoiding such edges in FSM_i is impossible, a matching $u_{i,j}$ transition in the IUT will be untestable. In spite of this restriction, initial research has shown [11] that this approach substantially increases the test coverage for several protocols. Examples are MIL-STD 188-220B and the IEEE 802.2 LLC Type 2 Connection Component.

With the controllability restriction and the augmentation of G with preambles and postambles (prior to converting G to G'), some of the possible interactions between the IUT and FSM_i are not modeled in G'. Since matching transitions are typically defined in most of the states of FSM_i, this divergence of G' from a minimal system representation is expected to be insignificant (for example, in 188-220B

Data Link Layer, every state in the Intranet Layer FSM_i has a matching transition for a number of semicontrollable inputs). Therefore, a minimum cost tour derived from G' (Section 6) can be claimed as a near-minimum cost test sequence of G.

8 Conclusion

In an embedded testing environment, where an IUT communicates with multiple entities, a tester may have differing degrees of control on the interactions between these entities and the IUT: directly controllable, semicontrollable, or uncontrollable. Semicontrollable and uncontrollable interactions severely reduce the testable portions of an SUT. In addition, race conditions may arise during testing due to multiple communicating interfaces.

While nothing can be done regarding uncontrollable interactions, semicontrollable inputs can be utilized to improve test coverage. This paper provides a practical algorithmic technique for test generation that utilizes as many indirectly controllable inputs as possible without creating nondeterministic behavior of the test system. Although, for the most general case, the graph conversion results in an exponentially large number of nodes, practical considerations can make the converted graph size feasible. The algorithm builds a transition graph for test derivation without explicitly constructing a composite global FSM modeling an SUT. This approach has a significant advantage, since an FSM_i's state and transition space may be large. For each FSM_i, only those transitions that can be used to test an IUT's transitions are considered during the graph construction.

This methodology has been applied to generate tests for MIL-STD 188-220B. By using the graph conversion and the race condition elimination approaches presented in this paper, the number of testable state transitions increased from approximately 200 to over 700, which represents an increase in a test coverage from 30% to 95% of the transitions defined in the 188-220B specification.

The extension of this work is planned to cover the embedded systems with multiple testers, which requires addressing synchronization issues in multi-party testing [34]. Also, more efficient algorithms for finding preambles and postambles that do not build a reduced composite machine will be investigated.

References

[1] AHO, A. V., DAHBURA, A. T., LEE, D., AND UYAR, M. U. An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours. *IEEE Trans. Commun. 39*, 11 (Nov. 1991), 1604–1615.

[2] ANSI/IEEE. *ISO/IEC International Standard 8802-2*, Dec. 1994.

[3] BUDKOWSKI, S., CAVALLI, A., AND NAJM, E., Eds. *Proc. IFIP Joint Int'l Conf. FORTE/PSTV* (Paris, France, Nov. 1998), Boston, MA: Kluwer Academic Publ.

[4] BUDKOWSKI, S., AND DEMBINSKI, P. An introduction to Estelle: A specification language for distributed systems. *Comput. Networks ISDN Syst. 14*, 1 (1991), 3–24.

[5] CAVALLI, A. R., FAVREAU, J. P., AND PHALLIPPOU, M. Standardization of formal methods in conformance testing of communication protocols. *Comput. Networks ISDN Syst. 29*, 1 (1996), 3–14.

[6] CORMEN, T. H., LEISERSON, C. E., AND RIVEST, E. L. *Introduction to Algorithms*. McGraw-Hill, New York, NY, 1992. $\surd$.

[7] DOD. *Military Standard—Interoperability Standard for Digital Message Device Subsystems (MIL-STD 188-220B)*, Jan. 1998.

[8] FECKO, M. A., AMER, P. D., SETHI, A. S., UYAR, M. U., DZIK, T., MENELL, R., AND MCMAHON, M. Formal design and testing of MIL-STD 188-220A based on Estelle. In *Proc. IEEE Milit. Commun. Conf. (MILCOM)* (Monterey, CA, Nov. 1997).

[9] FECKO, M. A., UYAR, M. U., AMER, P. D., AND SETHI, A. S. Using semicontrollable interfaces in testing Army communications protocols: Application to MIL-STD 188-220B. In *Proc. IEEE Milit. Commun. Conf. (MILCOM)* (Atlantic City, NJ, Nov. 1999).

[10] FECKO, M. A., UYAR, M. U., SETHI, A. S., AND AMER, P. D. Embedded testing in systems with semicontrollable interfaces. Tech. Rep. TR-98-18, CIS Dept., Univ. of Delaware, Newark, DE, 1998.

[11] FECKO, M. A., UYAR, M. U., SETHI, A. S., AND AMER, P. D. Issues in conformance testing: Multiple semicontrollable interfaces. In Budkowski et al. [3], pp. 111–126.

[12] HOPCROFT, J. E., AND ULLMAN, J. D. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[13] ISO, INFORMATION TECHNOLOGY—OSI. *ISO International Standard 9646: Conformance Testing Methodology and Framework*. Geneva, Switzerland, 1991.

[14] JTC1/SC21/WG1/PROJECT 54.1. *Framework: Formal Methods in Conformance Testing*, Feb. 1995.

[15] KIM, M., KANG, S., AND HONG, K., Eds. *Proc. IFIP Int'l Work. Test. Communicat. Syst. (IWTCS)* (Cheju Island, Korea, Sept. 1997), Boston, MA: Kluwer Academic Publ.

[16] LEE, D. Y., AND LEE, J. Y. A well-defined Estelle specification for the automatic test generation. *IEEE Trans. Comput. 40*, 4 (Apr. 1991).

[17] LENSTRA, J. K., AND RINNOOY KAN, A. H. G. On general routing problems. *Networks 6* (1976), 273–280.

[18] LIMA JR., L. P., AND CAVALLI, A. R. A pragmatic approach to generating test sequence for embedded systems. In Kim et al. [15].

[19] LINN, R. J. Conformance testing for OSI protocols. *Comput. Networks ISDN Syst. 18*, 3 (1990), 203–219.

[20] LINN, R. J., AND UYAR, M. U. *Conformance Testing Methodologies and Architectures for OSI Protocols*. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994. $\surd$.

[21] MILLER, R. E., AND PAUL, S. Structural analysis of protocol specifications and generation of maximal fault coverage conformance test sequences. *IEEE/ACM Trans. Network. 2*, 5 (Oct. 1994), 457–470.

[22] PETRENKO, A., AND YEVTUSHENKO, N., Eds. *Proc. IFIP Int'l Work. Test. Communicat. Syst. (IWTCS)* (Tomsk, Russia, Sept. 1998), Boston, MA: Kluwer Academic Publ.

[23] PETRENKO, A. F., AND YEVTUSHENKO, N. Fault detection in embedded components. In Kim et al. [15].

[24] PETRENKO, A. F., YEVTUSHENKO, N., AND V. BOCHMANN, G. Fault models for testing in context. In *Proc. IFIP Joint Int'l Conf. FORTE/PSTV* (Kaiserslautern, Germany, Oct. 1996).

[25] PETRENKO, A. F., YEVTUSHENKO, N., V. BOCHMANN, G., AND DSSOULI, R. Testing in context: Framework and test derivation. *Comput. Commun. 19*, 14 (1996), 1236–1249.

[26] PHALIPPOU, M. The limited power of testing. In *Proc. IFIP Int'l Work. Protocol Test Syst. (IWPTS)* (1992), Amsterdam: North-Holland.

[27] RAFIQ, O., AND CASTANET, R. From conformance testing to interoperability testing. In *Proc. IFIP Int'l Work. Protocol Test Syst. (IWPTS)* (Washington, DC, 1990), pp. 371–385.

[28] RAYNER, D. OSI conformance testing. *Comput. Networks ISDN Syst. 14*, 1 (1987), 79–98.

[29] SABNANI, K. K., AND DAHBURA, A. T. A protocol test generation procedure. *Comput. Networks ISDN Syst. 15* (1988), 285–297.

[30] SABNANI, K. K., UYAR, M. U., AND LAPONE, A. M. An algorithmic procedure for checking safety properties of communication protocols. *IEEE Trans. Commun. 37*, 9 (Sept. 1989), 940–948.

[31] TIMOHOVICH, E. An approach to protocol entity model development for embedded testing. *Automatic Control Comput. Sci. 27*, 3 (1993), 34–41.

[32] TRETMANS, J. Conformance testing with labelled transitions systems: Implementation relations and test generation. *Comput. Networks ISDN Syst. 29*, 1 (1996), 49–79.

[33] URAL, H. Formal methods for test sequence generation. *Comput. Commun. 15*, 5 (June 1992), 311–325.

[34] WU, W.-J., CHEN, W.-H., AND TANG, C. Y. Synchronizable test sequence for multi-party protocol conformance testing. *Comput. Commun. 21* (1998), 1177–1183.

[35] YEVTUSHENKO, N., CAVALLI, A. R., AND LIMA JR., L. P. Test suite minimization for testing in context. In Petrenko and Yevtushenko [22], pp. 127–145.

[36] ZENG, H. X., CHANSON, S. T., AND SMITH, B. R. On ferry clip approaches in protocol testing. *Comput. Networks ISDN Syst. 17*, 2 (1989), 77–88.

[37] ZHU, J., VUONG, S. T., AND CHANSON, S. T. Evaluation of test coverage for embedded system testing. In Petrenko and Yevtushenko [22], pp. 111–126.