OPTIMUM TEST SEQUENCE GENERATION FROM ESTELLE SPECIFICATIONS [*]

Mariusz A. Fecko, M. Ümit Uyar[†] Paul D. Amer, Adarshpal S. Sethi
Computer and Information Science Department
University of Delaware, Newark, DE

Abstract

The recent research results from the Protocol Engineering Lab of the University of Delaware in minimum-length test generation based on Estelle specifications are summarized. The methods use models restricting the number of self-loops to be consecutively traversed for each state, and utilizing semicontrollable interfaces of the IUT to increase the number of testable transitions.

The initial results show that test sequences can be successfully generated for real-life Estelle specifications, such as MIL-STD 188-220B. The test sequences are free of interruptions due to unexpected timeouts. Also, the number of testable transitions is increased by 300%.

1 Introduction

Conformance test generation techniques reported in literature [1] - [6], using a deterministic finite-state machine (FSM) model of a protocol specification, focus on the optimization of the test sequence length.

However, an Implementation Under Test (IUT) may have timing constraints imposed by active timers. If these constraints are not considered during test sequence generation, the sequence may not be realizable in a test laboratory. As a result, valid implementations may incorrectly fail the conformance tests.

Another problem in test sequence generation is due to the limited controllability of an IUT. Testers may not have direct access to all interface(s) in which the IUT accepts inputs. In this case, some of the inputs defined in the specification cannot be directly applied to an IUT (typically, the interfaces with upper layers or with timers). The interactions involving such interfaces may render some portions of the protocol untestable, and may introduce nondeterminism and/or race conditions during testing.

These timing and controllability issues were present in the extended FSM (EFSM) model of the Estelle [7, 8, 9] specification of MIL-STD 188-220B [10, 11, 12] for Combat Net Radio communication. By using the results of the research outlined in this paper, the test sequences generated for MIL-STD 188-220B are free of interruptions due to unexpected timeouts. The penalty paid for the realizable tests is greater test sequence cost. Also, the number of testable transitions increased from approximately 200 to over 700 by utilizing multiple interfaces without controllability conflicts.

Section 2 of the paper outlines the formulation of the minimum-cost test sequence generation under timing constraints. The controllability issues are discussed in Section 3, where a graph augmentation can fully utilize the semicontrollable interfaces to generate minimum-cost test sequences with increased coverage. Concluding remarks appear in Section 4.

2 Generation of realizable tests under timing constraints

During testing, traversing each state transition of an IUT requires a certain amount of time. A test sequence that traverses too many *self-loops* (a *self-loop* is a state transition that starts and ends at the same state) in a given state will not be realizable in a test laboratory if the time to traverse the self-loops exceeds a timer limit as defined by another transition originating in this state. In this case, a timeout will inadvertently trigger forcing the IUT into a different state, and thereby disrupting the test sequence before all of the self-loops are traversed. If this unrealizable test sequence is not avoided during test generation, most IUTs will fail the test even when they meet the specification. Clearly, this is not the goal of testing. Therefore, a properly generated test sequence must take timer constraints into account.

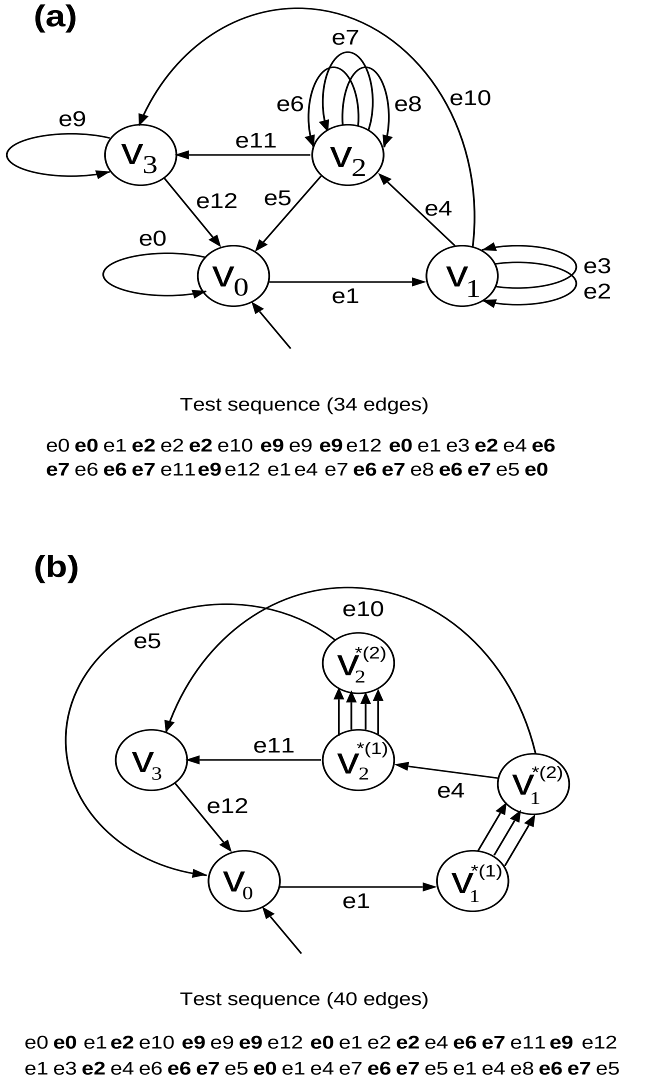
Examples of protocols that contain many self-loop transitions in their FSM models include ISDN Q.931 for supplementary voice services, MIL-STD 188-220B [10] for Combat Net Radio communication, and LAPD [13], the data link protocol for the ISDN's D channel.

In addition to the original self-loops of a specification model, additional self-loops are typically created when generated test sequences use state verification techniques such as unique input/output (UIO) sequences [14], distinguishing sequences [15, 16], or characterizing sequences [15, 16].

The presented methodology [17, 18, 19] optimizes the test sequence length and cost, under the constraint that an IUT can remain only a limited amount of time in some states during testing, before a timer's expiration forces a state change. The solution first augments an original graph representation of the protocol FSM model. Then it formulates a Rural Chinese Postman Problem solution [20] to generate a minimum-length tour. In the final test sequence generated, the number of consecutive self-loops never exceeds any state's specified limit. In most cases, this test sequence will be longer than one without the constraint since limiting the number of self-loop traversals likely re-

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

[†] Dr. Uyar, a Research Professor with Department of Electrical Engineering, the City College of the City University of New York, is presently Visiting Associate Professor at University of Delaware.



Test sequence (34 edges)

e0 **e0** e1 **e2** e2 **e2** e10 **e9** e9 **e9** e12 **e0** e1 e3 **e2** e4 **e6** **e7** e6 **e6** **e7** e11 **e9** e12 e1 e4 e7 **e6** **e7** e8 **e6** **e7** e5 **e0**

Test sequence (40 edges)

e0 **e0** e1 **e2** e10 **e9** e9 **e9** e12 **e0** e1 e2 **e2** e4 **e6** **e7** e11 **e9** e12 e1 e3 **e2** e4 e6 **e6** **e7** e5 **e0** e1 e4 e7 **e6** **e7** e5 e1 e4 e8 **e6** **e7** e5

Figure 1: Minimum-cost test sequence (a) without and (b) with self-loop repetition constraint.

quires additional visits to a state which otherwise would have been unnecessary.

The methodology uses UIO sequences for state verification. However, the results presented also are applicable to test generation that uses distinguishing or characterizing sequences. Earlier results of this study, limited to verification sequences that are self-loops, are presented in [19]. The later papers [17, 18] generalize these earlier results to both self-loop and non-self-loop verification sequences.

Example 1: Consider the FSM (represented by the graph $G(V, E)$) with self-loop transitions shown in Figure 1 (a). Suppose that vertices v_0, v_2, and v_3 of the FSM can tolerate at most three, and v_1 at most two self-loop transitions during each visit. Let transitions $e10$ and $e11$ correspond to timeouts. After either $e10$ or $e11$ is triggered, the FSM is brought into state v_3.

Let us define the following parameters for G:

- $d_{state\_ver}(v_i)$ - number of self-loops in the UIO sequence of (v_i);
- $max\_self(v_i)$ - the maximum number of self-loops in a test sequence that can be traversed during each visit to v_i. Attempting to remain in state v_i longer to execute more than $max\_self(v_i)$ self-loops would result in disruption of a test sequence;
- $d_{min\_self}(v_i)$ - the minimum number of times a tour covering all edges in E must include vertex $v_i \in V$ (the derivation of $d_{min\_self}(v_i)$ is shown in [17, 18, 19]).

UIO sequences and the values of max_self, d_{state_ver} and d_{min_self} for vertices v_0, v_1, v_2, and v_3 are as follows:

Vertex	*UIO*	max_self	d_{state_ver}	d_{min_self}
v_0	e0	3	1	2
v_1	e2	2	1	3
v_2	e6,e7	3	2	4
v_3	e9	3	1	2

The Chinese postman method [21] when applied to the graph without any self-loop repetition constraint results in the test sequence

$$\underline{e0, \mathbf{e0}, e1, \mathbf{e2}, e2, \mathbf{e2}, e10}, \mathbf{e9}, e9, \mathbf{e9}, e12, \mathbf{e0}, e1, e3, \mathbf{e2}, e4, \mathbf{e6}, \mathbf{e7}, e6, \mathbf{e6}, \mathbf{e7}, e11, \mathbf{e9}, e12, e1, e4, e7, \mathbf{e6}, \mathbf{e7}, e8, \mathbf{e6}, \mathbf{e7}, e5, \mathbf{e0} \tag{1}$$

containing 34 edges. Edges used for the purpose of state verification appear in bold.

As can be seen from the underlined part of the above test sequence, after $e1$ is traversed, the IUT should stay in state v_1 for a time that allows at least three self-loop traversals. However, this part of the test sequence is not realizable in a test laboratory because the timeout edge $e10$ will be triggered after the second consecutive self-loop traversal (i.e., $max\_self(v_1) = 2$). The IUT will prematurely move into v_3 and the test sequence will be disrupted.

To address the problem of test sequence disruption due to timeouts, the graph of Figure 1 (a) is converted to the graph shown in Figure 1 (b). Since in this example all UIO sequences are self-loops, the simplified conversion presented in [19] is sufficient. The vertices for which a premature timeout may disrupt a test sequence, which are v_1 and v_2, are split and then connected by $d_{min\_self}(v_1) = 3$ and $d_{min\_self}(v_2) = 4$ edges, respectively.

Considering the constrained self-loop problem, the test sequence for the graph of Figure 1 (b) is obtained as

$$e0, \mathbf{e0}, e1, \mathbf{e2}, e10, \mathbf{e9}, e9, \mathbf{e9}, e12, \mathbf{e0}, e1, e2, \mathbf{e2}, e4, \mathbf{e6}, \mathbf{e7}, e11, \mathbf{e9}, e12, e1, e3, \mathbf{e2}, e4, e6, \mathbf{e6}, \mathbf{e7}, e5, \mathbf{e0}, e1, e4, e7, \mathbf{e6}, \mathbf{e7}, e5, e1, e4, e8, \mathbf{e6}, \mathbf{e7}, e5 \tag{2}$$

containing 40 edges.

Although longer than that of Figure 1 (a), the test sequence in Figure 1 (b) is minimum-length with the introduced self-loop constraint. During each visit to vertices v_0, v_1, v_2 and v_3, the number of consecutive self-loop edges traversed is less than or equal to the maximum allowed number of self-loop traversals. Therefore, this test sequence is realizable in the test laboratory.

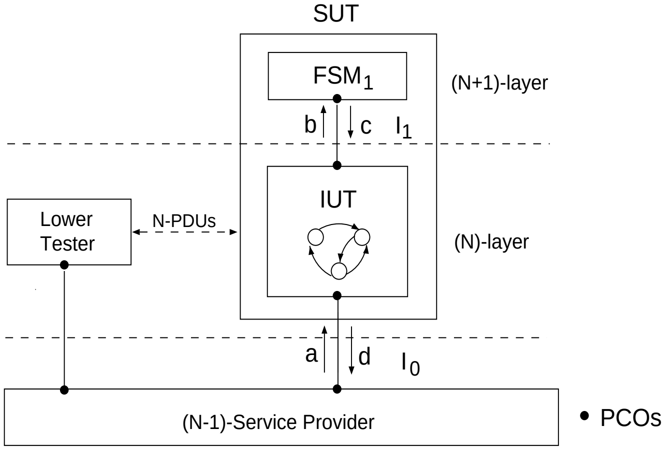


Figure 2: Testing (N)-layer IUT with an (N+1)-layer semicontrollable interface.

3 Test generation for systems with multiple semicontrollable interfaces

In the testing framework of Figure 2, the interface I_1 is not exposed in the SUT. In other words, the inputs from (N+1)-layer cannot be directly applied to the IUT, nor can the outputs generated by the IUT be observed at (N+1)-layer. Such an interface I_1 is called *semicontrollable* if FSM_1 can be utilized to supply inputs to the IUT. On the other hand, the tester can apply inputs to the IUT directly by using a lower tester, which exchanges N-PDUs with the IUT by using the (N-1)-Service Provider. The interface I_0 between the lower tester and the IUT is therefore *directly controllable.*

The transitions in IUT that require inputs from a semicontrollable interface I_1 can be tested only by utilizing one of the directly controllable interfaces (e.g., by supplying input a in Figure 2 to the IUT). The IUT responds with outputs (such as message b in Figure 2), which are applied to FSM_1 at I_1's points of control and observation (PCOs) [3, 22]. As response to those outputs, FSM_1 will send inputs (similar to message c in Figure 2) to the IUT through semicontrollable interface I_1. These inputs will trigger the appropriate transitions in the IUT, which will send output(s) to the lower tester (message d in Figure 2).

The methodology presented in [23] addresses the problem of generating optimal realizable test sequences in an environment with multiple semicontrollable interfaces. The methodology fully utilizes semicontrollable interfaces in an IUT while avoiding the race conditions. An algorithm is introduced in [23] to modify the directed graph representation of the IUT such that its semicontrollable portions become directly controllable, where possible. In the most general case, obtaining such a graph conversion may end up with exponentially large number of nodes. However, it is shown [23] that special considerations such as the small number of interfaces interacting with an IUT and diagnostics considerations make the problem size feasible for most practical cases.

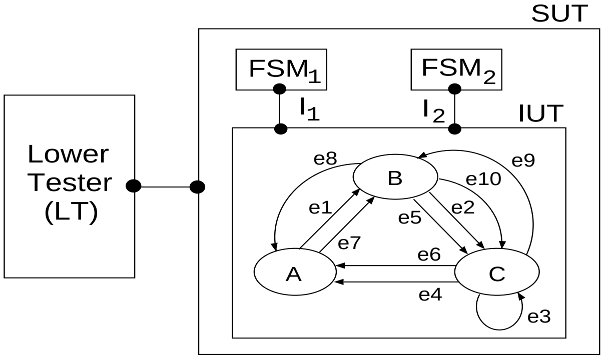


Figure 3: IUT interacting with two semicontrollable interfaces.

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

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.

Example 2: Consider the IUT of Figure 3 which is interacting with semicontrollable FSM_1 and FSM_2 through the semicontrollable interfaces I_1 and I_2, respectively. The IUT's FSM (represented by graph G) is described in Table 1. Transition $e1$, triggered by input x_1 from the lower tester, generates output $o_{1,1}$ to FSM_1. In response, FSM_1 sends input $a_{1,1}$ which triggers transition $e3$. (In general, $a_{i,j}$ is the expected response to $o_{i,j}$.) Transition $e2$, which is triggered by a lower tester's input x_2, outputs $o_{2,1}$ to FSM_2, which responds with input $a_{2,1}$ triggering $e4$. Then $e4$ outputs $o_{1,2}$ to FSM_1, which responds with $a_{1,2}$ triggering $e8$. On the other hand, transitions $e5$, $e6$, $e7$, and $e9$, can be triggered directly by the lower tester. $e6$, $e7$, and $e9$, do not generate outputs to the semicontrollable interfaces. $e5$ generates output $o_{2,2}$ to FSM_2, which does not send any input to the IUT.

After conversion (Figure 4), each state of G is replaced with at most four related states in $G^{'}$ corresponding to the buffer configurations 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. 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; the dashed-line edges are the ones that can be traversed only when either buffer

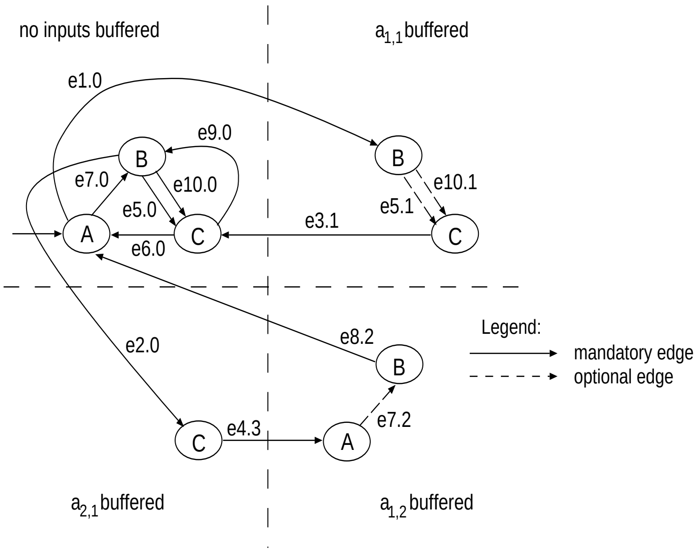


Figure 4: Graph transformation applied to the graph of Fig. 3. Mandatory and optional edges appear in solid and dashed lines, respectively.

contains an input. Due to the practical diagnostic considerations [23], we prefer testing edges when no inputs are buffered in semicontrollable interfaces. The Aho et al. [5] optimization technique gives the minimum-length test sequence for G' shown in Table 2. Steps with (→) indicate that an edge is tested in this step. Note that, for simplicity, the UIO sequences [14] are not included in this sequence.

4 Conclusion

This paper summarizes some recent results of generating minimum-length test cases from Estelle specifications under the constraints of timing and controllability.

The initial results of this research are being used to generate test sequences for MIL-STD 188-220B, which are free of interruptions due to unexpected timeouts, and increase the number of testable transitions by 300%.

Step	*Edge name*	*Input from*	*Output to*
→ 1	e1.0	$LT?x_1$	$FSM_1!o_{1,1}$
2	e5.1	$LT?x_5$	$FSM_2!y_5$
→ 3	e3.1	$FSM_1?a_{1,1}$	$LT!y_3$
→ 4	e6.0	$LT?x_6$	$LT!y_6$
→ 5	e7.0	$LT?x_7$	$LT!y_7$
→ 6	e2.0	$LT?x_2$	$FSM_2!o_{2,1}$
→ 7	e4.3	$FSM_2?a_{2,1}$	$FSM_1!o_{1,2}$
8	e7.2	$LT?x_7$	$LT!y_7$
→ 9	e8.2	$FSM_1?a_{1,2}$	$LT!y_8$
10	e7.0	$LT?x_7$	$LT!y_7$
→ 11	e5.0	$LT?x_5$	$FSM_2!y_5$
→ 12	e9.0	$LT?x_9$	$LT!y_9$
13	e5.0	$LT?x_5$	$FSM_2!y_5$
14	e6.0	$LT?x_6$	$LT!y_6$

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

References

[1] H. Ural and B. Yang, "A test sequence selection method for protocol testing," *IEEE Trans. on Communications*, vol. 39, no. 4, 1991.

[2] R. E. Miller and S. Paul, "On the generation of minimal-length conformance tests for communication protocols," *IEEE/ACM Trans. on Networking*, vol. 2, pp. 116–129, Feb 1993.

[3] B. S. Bosik and M. U. Uyar, "FSM-based formal methods in protocol conformance testing: from theory to implementation," *Computer Networks and ISDN Systems*, vol. 22, pp. 7–34, Sep 1991.

[4] W. Y. Chan and S. T. Vuong, "An improved protocol test generation procedure based on UIOs," in *Proc. ACM SIGCOMM*, Sep 1989.

[5] A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar, "An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours," *IEEE Trans. on Communications*, vol. 39, pp. 1604–1615, Nov 1991.

[6] B. Sarikaya, G. von Bochmann, and E. Cerny, "A test design methodology for protocol testing," *IEEE Trans. Software Engineering*, vol. 13, pp. 518–531, May 1987.

[7] Information Processing Systems - Open System Interconnection, *ISO International Standard 9074: Estelle - a Formal Description Technique Based on an Extended State Transition Model*, 1989.

[8] S. Budkowski and P. Dembinski, "An introduction to Estelle: A specification language for distributed systems," *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 3–24, 1991.

[9] R. Tenney, "A tutorial introduction to Estelle," Tech. Rep. 88–1, Univ of Mass, Boston, Jun 1988.

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

[11] M. A. Fecko, P. D. Amer, A. S. Sethi, M. U. Uyar, T. Dzik, R. Menell, and M. McMahon, "Formal design and testing of MIL-STD 188-220A based on Estelle," in *Proc. MILCOM'97*, (Monterey, California), Nov 1997.

[12] P. D. Amer, M. A. Fecko, A. S. Sethi, M. U. Uyar, T. J. Dzik, R. Menell, and M. McMahon, "Using Estelle to evolve MIL-STD 188-220," (in review).

[13] M. U. Uyar and M. H. Sherif, "Protocol modeling for conformance testing: Case study for the ISDN LAPD protocol," *AT&T Technical Journal*, vol. 69, Jan 1990.

[14] K. K. Sabnani and A. T. Dahbura, "A protocol test generation procedure," *Computer Networks and ISDN Systems*, vol. 15, pp. 285–297, 1988.

[15] A. Bhattacharyya, *Checking Experiments in Sequential Machines*. New York, N.Y.: John Wiley & Sons, 1989.

[16] Z. Kohavi, *Switching and Finite Automata Theory.* New York, N.Y.: McGraw Hill, 1978.

[17] M. U. Uyar, M. A. Fecko, A. S. Sethi, and P. D. Amer, “Generation of realizable conformance tests under timing constraints,” in *Proc. MILCOM'98*, (Bedford, MA), Oct 1998.

[18] M. U. Uyar, M. A. Fecko, A. S. Sethi, and P. D. Amer, “Test generation for protocols with timing constraints,” Tech. Rep. TR-98-07, CIS Dept., University of Delaware, Newark, DE, 1997. (Submitted for publication).

[19] M. U. Uyar, M. A. Fecko, A. S. Sethi, and P. D. Amer, “Minimum-cost solutions for testing protocols with timers,” in *Proc. IEEE Int'l Performance, Computing, and Communications Conf.*, (Phoenix, AZ), pp. 346–354, Feb 1998.

[20] J. K. Lenstra and A. H. G. R. Kan, “On general routing problems,” *Networks*, vol. 6, pp. 273–280, 1976.

[21] M. U. Uyar and A. T. Dahbura, “Optimal test sequence generation for protocols: the Chinese postman algorithm applied to Q.931,” in *Proc. IEEE GLOBECOM*, pp. 68–72, Dec 1986.

[22] ISO, Information Technology - OSI, Geneva, Switzerland, *ISO International Standard 9646: Conformance Testing Methodology and Framework*, 1991.

[23] M. A. Fecko, M. U. Uyar, A. S. Sethi, and P. D. Amer, “Issues in conformance testing: Multiple semicontrollable interfaces,” in *Proc. IFIP Joint Int'l Conf. FORTE/PSTV*, (Paris, France), Nov 1998.