TESTING CONFORMANCE TO ENERGY MANAGEMENT AND CONTROL SYSTEM COMMUNICATION PROTOCOLS—PART 2: TEST SUITE GENERATION

S.T. Bushby
Associate Member ASHRAE

ABSTRACT

ASHRAE has formed a committee to develop a standard communication protocol for energy management and control systems (EMCS). The goal of being able to connect control equipment from any vendor and make it work as part of an integrated system will not be achieved until tests to determine conformance to the standard are developed. This paper is the second in a two-part series addressing the question of testing conformance to an EMCS protocol.

This paper reviews international efforts to develop procedures for generating test suites used to determine conformance to a communication protocol standard. Four criteria are defined for evaluating the alternatives, and recommendations for the ASHRAE standard are made. An outline for the ASHRAE test suite is presented along with comments about the future steps needed to completely define it.

INTRODUCTION

In January 1987 ASHRAE formed Standards Project Committee 135P (SPC 135P) to undertake the task of developing a standard communications protocol for building energy management and control systems (EMCS) (Bushby and Newman 1988). The goal of SPC 135P is to develop an industry consensus standard, which, when implemented, will permit control devices made by any manufacturer who complies with the standard to be easily integrated into one control system. This goal will not be completely achieved until control devices can be tested to determine whether they conform to the standard.

The first paper in this series described the conformance testing process in general terms. Specific recommendations were made for a test system architecture suitable for testing conformance to the developing ASHRAE protocol standard. These recommendations were based on a review of the current standards, draft standards, and published literature on testing conformance to communication protocols.

Once an abstract test methodology has been accepted and some means of implementing it on a real system has been devised, the problem of creating the actual tests remains. This is a nontrivial problem and several approaches to dealing with it have been suggested in the literature. This paper will review the literature and propose an approach for developing the tests that would be used to determine conformance to the ASHRAE protocol. These tests collectively make up what is called a "conformance test suite."

THE CONFORMANCE TEST SUITE

The International Standards Organization (ISO) has developed draft proposals and draft international standards for testing conformance to standards for open system interconnection (ISO 1987c; 1988a, b, c). One of these draft standards addresses the issue of specifying abstract test suites (ISO 1988b). An abstract test suite is a hierarchical structure composed of test groups, test cases, test steps, and test events (ISO 1988a, b). The suite is abstract because the specifications are made in an implementation-independent manner. The actual code used to implement the tests is not specified by a standard.

Test cases fall in the middle of the hierarchy and are the key to understanding the structure. A test case has a narrowly defined purpose, such as verifying that the implementation under test (IUT) has one particular required capability. Test cases are combined into test groups, which provide a logical ordering, and all of the test groups combine to make up an abstract test suite.

A test case is composed of an ordered series of test steps. The test steps are typically divided into three parts: a preamble, the actual test, and a postamble. The preamble is one or more steps used to place the IUT in a required start state. This is followed by one or more steps to conduct the actual test. Finally, the postamble is used to return the IUT to a desired final state. Each test step consists of one or more test events. A test event is the transfer of a single protocol data unit (PDU) or access service primitive (ASP) to or from the IUT.

Steven T. Bushby is an Electronics Engineer in the Mechanical Systems and Controls Group, Building Environment Division, Center for Building Technology, National Institute of Standards and Technology, Gaithersburg, MD.

EVALUATING APPROACHES TO TEST SUITE DESIGN

Several approaches to designing conformance test suites have been tried. Each of these methods has its own strengths and weaknesses. It is important to establish some criteria for evaluating these different approaches, and the following four criteria will be used for this purpose:

1. Completeness—The test suite should contain a test case that corresponds in some manner to each conformance requirement of the standard. This includes optional variations permitted in the standard and any variations due to classes of conformance.
2. Practical to implement—The test suite should be small enough to be administered without placing a severe burden on either the tester or the protocol implementor.
3. Fair and reproducible—No advantage should be given to any particular implementation or implementation approach. Repeated testing of the same implementation should give the same result.
4. Semantic coupling between test cases and conformance requirements—Each test case should be designed to test a specific requirement of the standard. If a particular test case passes or fails, it should be clear which conformance requirement of the standard applies.

Approaches to test suite design can be broadly classified as manual or semiautomated. The semiautomated techniques attempt to take advantage of formal descriptions of the protocol to assist in test suite development. These approaches include the use of transition tours, the DS-method, the W-method, and generative grammars, all of which will be described and evaluated according to the criteria above.

MANUAL APPROACHES TO TEST SUITE DESIGN

Exhaustive testing of communication protocols is impossible because the set of all possible combinations of communication events is unbounded. An approach is needed that selects "interesting" test cases, which, when applied, will increase confidence that an implementation will perform correctly under a comprehensive set of circumstances. For most protocols, the design and selection of these test cases has been done manually (Hengeveld and Kroon 1987). The manual approach involves one or more persons, with detailed knowledge of the protocol specification, essentially thinking up relevant test cases and the specific test steps for each case. Conformance tests for recent protocols developed in this way include Manufacturing Automation Protocol (MAP) (Mathews et al. 1986; Muralidhar 1987), OSI Transport layer (Nightingale 1981, 1982), and Integrated Services Digital Network (Rathgeb 1987).

The main advantage of this approach over semiautomated techniques is the semantic coupling between test cases and conformance requirements, criterion 4 above. The precise intent of each test case can be clearly stated, and there is a direct correspondence between the results of the test and the conformance requirements of the standard. If a particular test is passed or failed, it is clear what this means in terms of meeting the conformance requirements of the standard.

Manual approaches can also do very well when considering the practicality of implementation. The designer has a great deal of control over the number of test cases in the suite and, by working carefully, can avoid many redundant tests. This can keep the test suite small enough to be practical. Also, an English language specification of the protocol is adequate.

From the standpoint of fairness and reproducibility, manual approaches do not particularly stand out as good or bad. Errors or omissions on the part of the designer could cause difficulty in this regard, but it is certainly possible to manually design a fair and reproducible test suite.

There is an obvious disadvantage to this approach and that is ensuring completeness. The test suite may have incomplete coverage because there may be a number of tests the test designer does not think of or that may be intuitively unappealing. Tests that could lead to significantly greater confidence in the reliability of the IUT might be overlooked. This problem is significant, but it tends to be self-correcting over a period of time. Experience with testing a protocol can lead to new insights that fill in the missing pieces.

SEMIAUTOMATED APPROACHES TO CONFORMANCE TESTING

Semiautomated approaches to test suite generation are an attempt to overcome the principal shortcoming of manual approaches, which is the inability to guarantee completeness. Most of these approaches are based on analysis and testing of finite state machine (FSM) models and apply techniques developed for fault detection in electronic circuits. They require either a formal specification of the protocol based on finite state machines or a finite state machine model developed from the informal protocol specification.

The basic approach is to generate a sequence of valid inputs to the FSM, in this paper called a checking sequence. This sequence is then used to exercise the IUT. The output from the IUT is "checked" against the output defined for the FSM and, if they agree, the IUT passes the test. The methods vary in the means used to generate the checking sequence and the type of protocol errors they can detect. Several different methods for generating checking sequences have been proposed, including transition tours, the DS-method, the W-method, and generative grammars.

Transition Tour Method

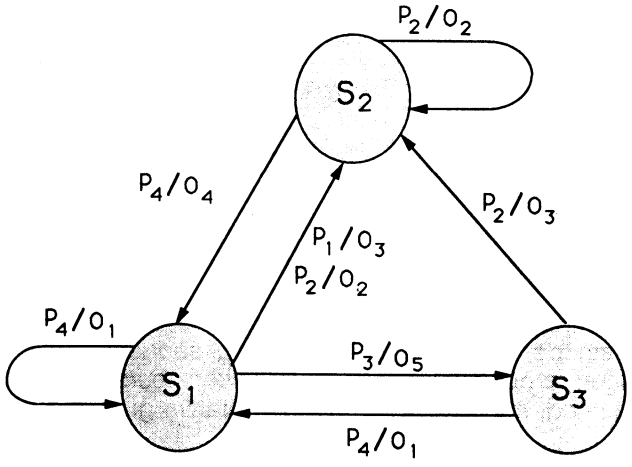
Consider a graph with nodes representing the states of the FSM and edges representing the transitions between states. For a communication protocol, the transitions take place based on protocol events. It is often the case that more than one protocol event will cause the same transition. A transition tour is an input sequence, starting with some initial state, which covers all transitions defined in the FSM (Bochman and Sarikaya 1982; Favreau and Linn 1987; Naito and

Tsunoyama 1981; Sarikaya and Bochman 1984). For transitions that can be fired by more than one input event, a transition for each possible event is included in the tour. The transition tour sequences may be generated by modifying graph traversal algorithms (Sarikaya and Bochman 1984). An example of a simple FSM and one of its transition tours is shown in Figure 1. It has been shown that transition tours detect all errors in the output function but may not detect all errors in the next state function (Sarikaya and Bochman 1984).

The DS-Method

The DS-method is based on the concept of a distinguishing sequence (DS). A DS is an input sequence that generates a unique output for each state of the FSM. That is, if the FSM model has n states, there exists a finite input sequence for which the FSM produces n different responses, according to which state it is in when the sequence is applied. By applying the DS and observing the results, it is possible to determine the state a FSM was in before the DS was applied. The state of the machine after applying the DS may or may not be the same as it was before. Not all FSM have a DS. An algorithm for generating a DS may be found in circuit theory textbooks (Friedman and Menon 1975, p. 143). An example FSM and its two DS are shown in Figure 2.

Several methods for generating a checking sequence based on the use of a DS exist. The original version, applied to communication protocols, was developed by Gonec (1970). The checking sequence for this approach is made up of two parts called the α sequence and the β sequence. The α sequence is used to verify that the IUT implements all of the states of the FSM and that they accept the DS. The β sequence is used to traverse all of the possible transitions in the FSM. Algorithms for generating the α and β sequences may be found in Gonec (1970).

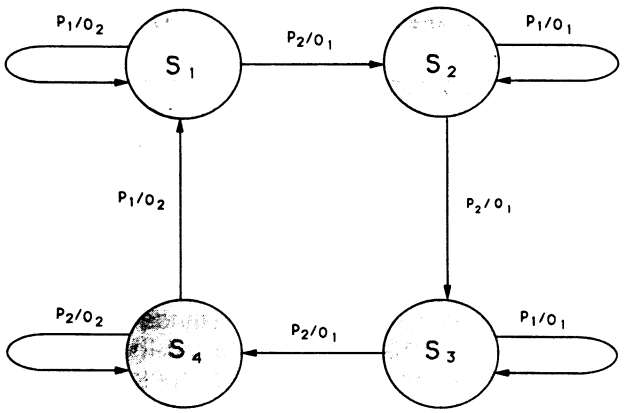


(a) An example finite state machine notation: P_i/O_j means the transition occurs on the input service primitive P_i and produces output O_j.

$S_1\ P_1\ S_2\ \ P_4\ S_1\ \ P_2\ S_2\ \ P_4\ S_1\ \ P_4\ S_1\ \ P_3\ S_3$
$P_4\ S_1\ \ P_3\ S_3\ \ P_2\ S_2\ \ P_2\ S_2$

(b) One possible transition tour
notation: start-state input final-state input final-state . . .

Figure 1



notation: P_i/O_j means the transition occurs on the input service primitive P_i and produces output O_j.

Initial State	Output Sequence Generated by Input Sequence $P_2P_2P_2$	Output Sequence Generated by Input Sequence $P_2P_2P_1$
S_1	$O_1O_1O_1$	$O_1O_1O_1$
S_2	$O_1O_1O_2$	$O_1O_1O_2$
S_3	$O_1O_2O_2$	$O_1O_2O_2$
S_4	$O_2O_2O_2$	$O_2O_2O_2$

Figure 2 *An example FSM with two distinguishing sequences, $P_2P_2P_2$ and $P_2P_2P_1$*

The DS-method requires that the FSM be fully connected and completely specified (Chow 1978). These are important restrictions because FSM for many real protocols cannot meet them (Aho et al. 1988; Bremer et al. 1984; Favreau and Linn 1987). When this method can be applied, the resulting test sequences tend to be extremely long (Aho et al. 1988). Some more recent modifications to this approach have been made that relax the definition of a DS so that the technique can be applied to a wider range of FSM (Aho et al. 1988; Hsieh 1971; Sabrani and Dahbura 1988). In one case (Aho et al. 1988), substantial reduction in the length of the generated sequences is also reported. The DS-methods have better error-detection capabilities than the transition tour method. All errors in the real implementation of the FSM can be detected so long as no additional states have been added (Friedman and Menon 1971; Sarikaya and Bochman 1984).

The W-Method

The W-method, also called the PW-method or characterization sequence, was originally proposed by Chow (1978). Many authors have reported using this approach for generating test sequences for communication protocols (Bochman and Sarikaya 1982; Bremer et al. 1984; Favreau and Linn 1987; Sarikaya and Bochman 1984). This method can be applied to some FSM that do not have a DS. The FSM in Figure 1 is an example. There is no DS for this FSM; however, there is a set of inputs that can be used to "characterize" the states of the machine. This characterization set, W, is $\{P_1, P_4\}$. The key point is that this is not an

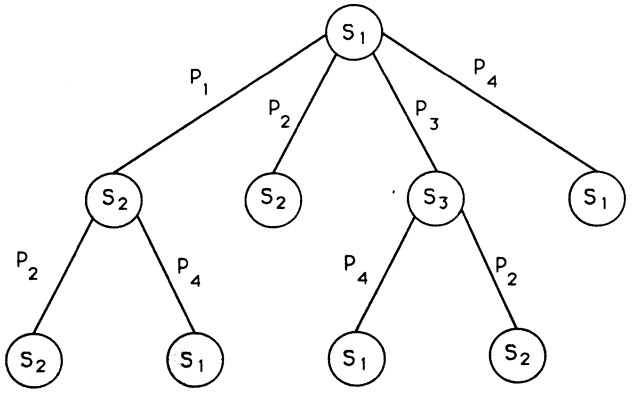
input sequence. The state changes that would occur for the input are ignored. Only the output resulting when each input from the set W is applied to a given state is considered. If the output set is unique for each state in the FSM, the input set "characterizes" the states of the machine. Table 1 illustrates the output set corresponding to $W = \{P_2, P_4\}$ for the FSM in Figure 1.

TABLE 1
State Responses to the Characterization Set $\{P_2, P_4\}$

state	Input P_2	Input P_4
1	O_2	O_1
2	O_2	O_4
3	O_3	O_1

The checking sequence is obtained by concatenation of two sets P and W ($P \cdot W$). The set P is the set of all partial paths in a test tree for the FSM, including the empty sequence. An algorithm for generating test trees is presented in Chow (1978). The set W is the characterization set described above. To conduct the test, each sequence from the concatenation of P and W is applied, beginning with the initial state. After a particular sequence is completed, the FSM is returned to the initial state to begin the next sequence. An example of a test tree and the set of partial paths is shown in Figure 3. The checking sequences resulting from $P \cdot W$ are:

P_2	P_3P_2	$P_1P_4P_2$
P_4	P_3P_4	$P_1P_4P_4$
P_1P_2	P_4P_2	$P_3P_4P_2$
P_1P_4	P_4P_4	$P_3P_4P_4$
P_2P_2	$P_1P_2P_2$	$P_3P_2P_2$
P_2P_4	$P_1P_2P_4$	$P_3P_2P_4$



(a) Testing tree for the finite state machine in Figure 1.

{}	
P_1	P_1P_2
P_2	P_1P_4
P_3	P_3P_4
P_4	P_3P_2

(b) Partial paths in the testing tree.

Figure 3

The characterization sequence approach requires that the FSM be fully connected and completely specified (Chow 1978). These are important restrictions because FSM for many real protocols cannot meet them (Aho et al. 1988; Bremer et al. 1984; Favreau and Linn 1987). When this method can be applied, the resulting test sequences tend to be extremely long (Aho et al. 1988). It has the same error-detection capability as the DS-method (Sarikaya and Bochman 1984).

Generating Test Sequences from Generative Grammars

Another approach to automating test sequence generation has been the use of generative grammars (Linn and McCoy 1983); readers not familiar with the terminology of BNF grammars should refer to Appendix B. In this approach, the input and output events of the protocol are considered to be the terminal symbols of a grammar. The nonterminal symbols are the possible states of the FSM. The productions of the grammar are constructed so that the language of the grammar describes the valid exchanges of protocol data units between the tester and the IUT.

At any point in a derivation, there may be several productions that could be used. A probability is assigned to each production, and a decision is made based on a random number generator. The test designer can manipulate the outcome of the sequence generation by changing the probabilities. This enables the designer to account for classes of conformance by eliminating productions that do not apply to a given conformance class. The authors reported that use of the random number generator resulted in many duplicate test sequences. These were removed by sorting techniques (Linn and McCoy 1983). When this was completed, the number of test sequences generated was reduced compared with the methods described above.

EVALUATION OF FSM METHODS FOR SEQUENCE GENERATION

The appeal of semiautomated sequence generation is that it might solve the problem of ensuring complete coverage of the protocol conformance requirements. It could be argued that the techniques described here are an improvement over manual approaches, but it is clear that they are not a solution to the problem. The difficulty is that, in most cases, an unmanageably large number of sequences are generated. A means of reducing the repetitive or semantically equivalent cases is required to make the test practical to implement. Techniques for doing this have been proposed (Favreau and Linn 1987), but they require subjective judgment on the part of the test suite designer. The subjective element that caused the problem in the first place is thus returned to the process in slightly different form.

Semantic coupling of test cases to conformance requirements, the principal strength of manual approaches, has been lost in these semiautomated techniques. The test cases are generated from the structure of the FSM model. To match the resulting

sequences to one or more conformance requirements requires human analysis. This increases the difficulty of deciding what the results of a particular test mean in terms of conforming to the standard.

Another problem with sequence generation based on finite state machines is that a classic finite state machine is not a good model for communication protocols. The problem is that state transitions in a real protocol depend on more than just the input and the current state. Formal description techniques like ESTELLE (ISO 1987a) have solved this problem by defining extended finite state machines. The difference is the addition of predicates, which are another factor in deciding when to fire state transitions. The test sequence generation methods described above are not based on extended FSM. Assumptions must be made about the value of the predicates used in the formal description in order to apply the sequence generation techniques at all. A discussion of this problem may be found in Favreau and Linn (1987). This introduces another subjective factor and contributes to the problem of large numbers of sequences because the generation may be repeated for several variations in the assumptions.

Another danger in using automated sequence generation based on FSM is that the model may represent the internal behavior of an idealized implementation, rather than the externally observable behavior of a system. The test sequences generated could, in some cases, become a test for conformance to the model and not the requirements of the standard. This issue is raised in Appendix A of Part 2 of the draft international standard on conformance testing in the context of overspecifying conformance requirements when formal descriptions are used as part of a standard (ISO 1988b). If a formal description overspecifies conformance requirements, then, clearly, test sequences generated from that description may be overly restrictive as well. This could raise legitimate questions about the fairness of the test.

The appeal of automated test sequence generation is great, which explains the large number of people who have been attempting to develop such techniques in recent years. Thus far, a fully successful approach has not been found, but it is an area worthy of future research. As a practical matter, there seems to be no advantage to using an automated approach to sequence generation for the ASHRAE protocol at this time.

RECOMMENDATIONS FOR THE ASHRAE PROTOCOL

Based on analysis of the current state of the art in automatic test sequence generation, it is recommended that a manual approach be used to develop the test sequences for the ASHRAE protocol. The use of semiautomated approaches is not sufficiently mature to be the basis for a standard. This conclusion seems to be supported by the ISO committee developing the conformance testing standard. Part 2 of the draft international standard (ISO 1988b) explicitly states that the relationship between abstract test suite specification and formal description techniques is outside the scope of the standard. A note further states that it is expected that an addendum to the standard will be issued in the future to change the scope and address this issue. This addendum is only in a very early stage of development (ISO 1987c).

Part 2 of the draft international standard (ISO 1988b) gives detailed recommendations about preparing the abstract test suite in clauses 8-11. These recommendations are a helpful guide to the test suite designer, and their use will increase the probability of complete coverage of the protocol conformance requirements.

Figure 4 shows a proposed outline of an abstract test suite for the ASHRAE protocol. This outline is a hierarchy of test groups. Much detail needs to be added to this outline to include the individual test cases, their purpose, and the test steps to be followed for each. At this level of detail, no tests for timing requirements or flow control have been included. Some portions of the standard are sufficiently developed to permit including this additional detail, but it is beyond the scope of this paper. A more detailed example test case, which illustrates the idea, can be found in Appendix A.

An important step in fleshing out the test suite is enumerating all of the conformance requirements for each clause in the standard. This is not only important for refining the test suite but also assists the standard committee in defining precisely what the conformance requirements are.

Four broad groups of tests are proposed. It is important to note that although they are logically distinct, there will, in fact, be overlap with regard to the actual test cases. One example of this is a series of test cases to determine that all of the required object types are supported. This can be done by repeated use of the ReadProperty service. These same tests can serve to verify that the ReadProperty service behaves correctly.

This overlap also illustrates an important point about analyzing the test results. A failure to return the requested properties of an object could occur because the object is not supported or because the ReadProperty service is not functioning properly. It may be that several tests must be looked at as a group to determine which of the logical test cases failed.

As it is currently envisioned, the ASHRAE protocol devices will be highly dependent on their configuration. For example, in most cases it will not be possible to create or delete objects on the fly. These objects will be defined when the controller is configured. This is important when testing conformance because a representative sample of objects must be available to conduct the conformance tests. This will require establishing a standard configuration, based on conformance class, which will be used for test purposes. These standard configurations must also account for the various options that might be supported in any particular implementation.

SUMMARY AND FUTURE DIRECTIONS

An abstract test suite as defined in draft international standards for testing conformance to com-

OUTLINE FOR ASHRAE PROTOCOL ABSTRACT TEST SUITE

A. Capability Tests
These tests will address the static conformance requirements of the standard and will focus on testing support for the required and optionally selected object-types and their properties.
A.1 Mandatory Object-Types
A.1.1 Conformance Class 1
:
:
A.1.n Conformance Class n
A1.2 Optional Object-Types
B. Behavior Tests: responses to valid behavior by peer implementation.
B.1 Application Services Initiated by Tester
B.1.1 Alarm and Event Services
B.1.2 File Access Services
B.1.3 Object Access Services
B.1.4 Remote Device Management Services
B.1.5 Virtual Terminal Services
B.2 Application Services Initiated by Implementation Under Test
B.2.1 Alarm and Event Services
B.2.2 File Access Services
B.2.3 Object Access Services
B.2.4 Remote Device Management Services
B.2.5 Virtual Terminal Services
C. Behavior Tests: response to syntactically invalid behavior by peer implementation.
D. Behavior Tests: response to syntactically correct but inopportune events by peer implementation.

Figure 4

munication protocol standards was described. A set of criteria for evaluating approaches to generating the test suite was established, and several methods were evaluated according to those criteria. All of the techniques discussed have shortcomings. Semiautomated approaches based on formal descriptions of the protocol are appealing but not sufficiently mature at this time to be viable for developing a conformance test for the ASHRAE protocol. A manual approach closely following the guidelines presented in the ISO draft international standard for abstract test suite specification is recommended.

A rough outline for an abstract test suite for the ASHRAE protocol was presented with comments on necessary refinements. These refinements should be made continuously as the standardization process progresses. This will assist the committee in making the conformance requirements of the standard clear and enable a suitable conformance test suite to be ready for use soon after the draft standard is complete.

ASHRAE is in a fortunate position because the experience gained and mistakes made in developing other communication protocol standards can be assimilated while the standard is still being developed. Coupling the development of conformance tests to the development of the protocol is an important part of this process.

REFERENCES

Aho, A.V.; Dahbura, A.T.; Lee, D.; and Uyar, M.U. 1988. "An optimization technique for protocol conformance test generation based UIO sequences and rural Chinese postman tours." *Proceedings of the IFIP WG 6.1 Eighth International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, VIII.* North-Holland, pp. 75-86.

Bochman, G.V., and Sarikaya, B. 1982. "Some experience with sequence generation for protocols." *Proceedings of the IFIP WG 6.1 Second International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, II.* North-Holland.

Bremer, J.; Mondvai, G.; Tarnay, K.; and Tibor, T. 1984. "Some experiences with test sequence generation in application layer." *Proceedings of the IFIP WG 6.1 Fourth International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, IV.* North-Holland, pp. 623-636.

Bushby, S.T., and Newman, H.M. 1988. "Standardizing energy management and control system communication protocols." *ASHRAE Journal,* Vol. 31, No. 1, January, pp. 33-36.

Chow, T.S. 1978. "Testing software design modeled by finite-state machines." *IEEE Trans. Software Engineering,* Vol. SE-4, No. 3, pp. 178-187.

Favreau, J.P., and Linn, R.J. 1987. "Automatic generation of test scenario skeletons from protocol specifications written in ESTELLE." *Proceedings of the IFIP WG 6.1 Seventh International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification,* VII. North-Holland, pp. 191-202.

Friedman, A.D., and Menon, P.R. 1971. *Fault detection in digital circuits,* p. 100. New York: Prentice-Hall.

Friedman, A.D., and Menon, P.R. 1975. *Theory and design of switching circuits.* Computer Science Press.

Gonec, G. 1970. "A method for the design of fault detection experiments." *IEEE Transactions on Computers,* Vol. C-19, No. 6, pp. 551-558.

Hengeveld, W., and Kroon, J. 1987. "Using checking sequences for OSI session layer conformance testing." *Proceedings of the IFIP WG 6.1 Seventh International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, VII.* North-Holland, pp. 435-449.

Hsieh, E.P. 1971. "Checking experiments for sequential machines." *IEEE Transactions on Computers,* Vol. C-20, No. 10, pp. 1152-1166.

ISO. 1987a. ISO Draft International Standard 9074, "Information processing systems—open systems interconnection—ESTELLE—a formal description technique based

on an extended state transition model." Available from ANSI, 1400 Broadway, New York, NY 10018.

ISO. 1987b. ISO Standard 8824, "Information processing systems—open systems interconnection—specification of abstract syntax notation one (ASN.1)." Available from ANSI, 1400 Broadway, New York, NY 10018.

ISO. 1987c. SC 21/N2001, "Working draft of OSI conformance testing methodology and framework, addendum to part 2: testing and FDTs." International Standards Organization. Available from ANSI, 1400 Broadway, New York, NY 10018.

ISO. 1988a. ISO DIS 9646-1, "Information processing systems—open system interconnection—OSI conformance testing methodology and framework—part 1: general concepts." International Standards Organization. Available from ANSI, 1400 Broadway, New York, NY 10018.

ISO. 1988b. ISO DIS 9646-2, "Information processing systems—open system interconnection—OSI conformance testing methodology and framework—part 2: abstract test suite specification." International Standards Organization. Available from ANSI, 1400 Broadway, New York, NY 10018.

ISO. 1988c. ISO Draft Proposal 9646-4, "Information processing systems—open system interconnection—OSI conformance testing methodology and framework—part 4: test realization." International Standards Organization. Available from ANSI, 1400 Broadway, New York, NY 10018.

Linn, R.J., and McCoy, W.H. 1983. "Producing tests for implementations of OSI protocols." *Proceedings of the IFIP WG 6.1 Third International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, III.* North-Holland, pp. 505-520.

Mathews, R.S.; Muralidhar, K.H.; and Schumacher, M.K. 1986. "Conformance testing: operational aspects, tools, and experiences." *Proceedings of the IFIP WG 6.1 Sixth International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, VI.* North-Holland, pp. 135-149.

Muralidhar, K.H. 1987. "MAP 2.1 network management and directory services test system." *Proceedings of the IFIP WG 6.1 Seventh International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, VII.* North-Holland, pp. 359-369.

Naito, S., and Tsunoyama, M. 1981. "Fault detection for sequential machines by transition-tour." *Proceedings, IEEE Fault Tolerant Computer Conference.*

Nightingale, J.S. 1981. "A benchmark for implementations of the NBS transport protocol." National Bureau of Standards Report No. ICST/HLNP 81-20, September.

Nightingale, J.S. 1982. "Protocol testing using a reference implementation." *Proceedings of the IFIP WG 6.1 Second International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, II.* North-Holland, pp. 513-520.

NIST. 1988. "A test system for implementations of FTAM/FTP gateways: final report: part 3: annex 8, test cases." Report No. ICST/SNA—88/6. Gaithersburg, MD: National Institute of Standards and Technology.

Rathgeb, E.P. 1987. "Protocol testing for the ISDN D-channel network layer." *Proceedings of the IFIP WG 6.1 Seventh International Workshop on Protocol Specification, Testing, and Verification. Protocol Specification, Testing, and Verification, VII.* North-Holland, pp. 421-434.

Sabnani, K., and Dahbura, A. 1988. "A protocol test generation procedure." *Computer Networks and ISDN Systems,* pp. 285-297.

Sarikaya, B., and Bochman, G.V. 1984. "Synchronization and specification issues in protocol testing." *IEEE Transactions on Communications,* Vol. Com-32, No. 4, April, pp. 389-395.

APPENDIX A
A Sample Conformance Test Case

This appendix is an example of a more detailed description of one test case that might be included in the test suite. It would be included in section A.1 of the outline in Figure 4. The purpose of this test case is to verify the support of a mandatory object type called the "EMCS device." This particular object type will be included in the standard and was chosen for this example because its properties are, for the most part, self explanatory. No knowledge of the details of the SPC 135P deliberations or a detailed specification of the object type are needed to understand what is intended.

Most objects will have at least some properties that can be written by remote devices. Testing support for these objects will, in part, require writing and then reading several values to each of these properties to be sure they are supported over the entire range specified in the standard. The Device Object Type is different in that its purpose is to convey information about the configuration of the device. To test for support of this object type, it is only necessary to read all of the properties and ensure that each response is valid and consistent with information provided in the protocol implementation conformance statement (PICS). The PICS describes the portions of the standard with which the implementation claims to conform and which allowed options have been selected.

Assume the object has the following properties:

Property	*Data Type (value)*
Property: Object_Identifier	EMCSObjectIdentifier ("VAV Ctrl 101")
Property: Object_Type	EMCSObjectType (EMCS_DEVICE)
Property: System_Status	EMCSDeviceStatus (OPERATIONAL)
Property: Vendor_Name	CharacterString ("XYZ Controls")
Property: Model_Name	CharacterSrring ("VAV 100")
Property: Firmware_Revision	CharacterString ("1.0")
Property: Application_Software_Version	CharacterString ("2-1-88")
Property: Location	CharacterString ("Room 101")
Property: Description	CharacterString ("")
Property: Protocol_Version	CharacterString ("1.0")
Property: Protocol_Conformance_Class	Integer (2)
Property: Protocol_Services_Supported	Set of EMCSService (ReadProperty, WriteProperty . . .)
Property: Protocol_Object_Types_Supported	Set of EMCS ObjectType (EMCS_DEVICE, ANALOG_INPUT, ANALOG_OUTPUT, BINARY_INPUT, BINARY_OUTPUT, LOOP)
Property: Max_Message_Length_Supported	Integer (32)
Property: Window_Size	Integer (1)

TEST A.1.1
Test Purpose:
Verify that the implementation supports the required object type "Device Object" and that the values of the properties are consistent with the PICS.
Test Description:
The conformance tester sends a ReadProperty request PDU to the IUT in an attempt to read all of the properties of the "Device Object." The response is analyzed to determine consistency with the standard and the PICS.
Expected Result:
A successful attempt to read all of the properties of the "Device Object." The values returned agree with the PICS and all are within the range specified by the standard.
Conformance Tester:
Issue a ReadProperty service request to the IUT requesting the value of all properties of the Device Object and specifying that the read access specifications (the names of the properties) be returned with the result.
IUT:
Process the ReadProperty service indication and issue a response containing the requested information.
Test Responder:
No action required in this test case.

It may be desirable to add a formalized description of what is expected here using Abstract Syntax Notation One (ISO 1987b). The details of how to do this are beyond the scope of this paper. The information shown here would become comments, and the formal description would make up the actual test case specification. This approach has been used recently for testing gateways for electronic mail systems (NIST 1988).

APPENDIX B
BNF Grammar Notation

Computer languages are typically specified using a formal notation for the grammar called Backus-Naur-Form. The formal specification is made up of three parts: terminal symbols, nonterminal symbols, and productions. The terminal symbols can be thought of as the letters of the alphabet in which the language is written. The nonterminal symbols are place holders, which can represent zero or more terminal symbols in a sentence of the language. The productions define the rules for substituting terminal symbols for the nonterminal symbols in an expression. As a set, the productions define all of the possible sentences in the language. A simple example may help make these ideas clear.

Consider a language made up of the terminal symbols (alphabet) {a, b, c, d} and the nonterminal symbols {S, A, B}. Further assume that the productions are:

$S \Rightarrow AaB$
$S \Rightarrow B$
$A \Rightarrow bc$
$A \Rightarrow Ad$
$B \Rightarrow c$

The production symbol, $\Rightarrow$, means that the nonterminal on the left-hand side may be replaced by the string of terminals and nonterminals on the right-hand side. Beginning with the start symbol, S, successive substitutions are made following the production rules until no more nonterminal symbols remain. This successive substitution is called a derivation. The result is a string or sentence in the language. Examples of some derivations and their resulting strings for this language are:

$S \Rightarrow B \Rightarrow c$
$S \Rightarrow AaB \Rightarrow AdaB \Rightarrow bcdaB \Rightarrow bcdac$
$S \Rightarrow AaB \Rightarrow bcaB \Rightarrow bcac$
$S \Rightarrow AaB \Rightarrow AdaB \Rightarrow AddaB \Rightarrow bcddaB \Rightarrow bcddac$

Typically the number of possible sentences is unbounded.

More information about the theory of computer languages can be found in textbooks on language theory or compiler writing.

DISCUSSION

B. Sun, Flack & Kurtz Engineers, San Francisco, CA: Will the EMS conformance testing be a part of the ASHRAE EMS Protocol Standard?

S.T. Bushby: It is not yet clear is the conformance tests will become part of Standard 135. It might make sense as a matter of convenience to ask the ASHRAE Standards Committee to allow us to publish the conformance tests as a separate standard. Either way, I do expect the conformance testing process to be included in an ASHRAE standard.