FAULT DIAGNOSIS UNDER TEMPORAL CONSTRAINTS

John W. Sheppard and William R. Simpson

ARINC Research Corporation, 2551 Riva Road, Annapolis, MD 21401

ABSTRACT

Currently, intelligent diagnostic systems are applied to problems in which the state of the system is fixed. Unfortunately, with systems increasing in complexity at high rates coupled with architectures promoting concurrence and fault tolerance, diagnosis in a temporal context is becoming imperative. In this paper we describe an approach to associating temporal knowledge with a diagnostic system using the propositional calculus to represent temporal relationships that allows efficient propagation of time constraints through the knowledge network. We conclude the paper with a brief discussion of how this work can be applied to the diagnostic domain.

INTRODUCTION

Research in artificial intelligence has advanced to the point where we can develop software capable of diagnosing complex systems at whatever level of detail required. The most common forms of artificial intelligence systems include rule based approaches, in which *if then* structures are used to describe the diagnostic problem, and model based systems, in which either the structure or the behavior of the system is represented mathematically to facilitate efficient and effective fault diagnosis. Several different problems relating to fault diagnosis concern representation of information and knowledge about the system to be diagnosed. A significant representation problem relates to how one represents information about time constraints in relation to the diagnostic situation.

Typically reasoning systems operate on an instantiation or a "snapshot" of some problem domain and deal specifically with logical relations between facts in the knowledge base. These logical relations often take the form of production rules that define linkages in a knowledge network. Yet typically these knowledge networks omit information about time. Specifically, some of the concerns relating to temporal knowledge include efficiently ordering tests so that various time constraints can be met. For example, suppose a piece of test equipment has a video display that requires a warming up period. If some tests can be performed immediately without the display, the desirability of performing these tests while we are waiting increases. This construct would be typically given as Test Group A cannot be performed prior to x minutes. Other constructs would deal with Test Group A before/after/during Test Group B and Test Group A within/before/after time interval y.

When considering the task of reasoning about time, two problems arise. The first is the problem of representing the temporal information. What is the primitive unit of time? How do these primitives combine to form an event? How do temporal events relate to one another? How are events and their interrelationships represented in a computer? These questions are fundamental to the problem of developing a temporal knowledge base. Once we determine initial interrelationships between events, we can pursue the propagation of constraints imposed by these events and their basic interrelationships throughout the network. This is the second problem and is referred to as constraint propagation. Once we determine a method for representing time and temporal relationships, together with an algorithm for performing the constraint propagation, we can construct a knowledge network of temporal events for an inference system such as a diagnostic engine.

REPRESENTING TEMPORAL INFORMATION

Primitives used in representing time events are either time points or time intervals. If we assume that points in time are primitive, then we can combine the points into a time interval with the end points delimiting the interval. On the other hand, many have felt that time interval is more reasonable primitive unit. In this case, the primitive is the

CH3148-4/92/0000-0151 $1.00©1992 IEEE

event. There is some disagreement among researchers over which primitive is best. We will therefore begin with a brief discussion of each type of representation.

Point Based Representations

If we assume the time point is primitive, then we can define an interval of time to consist of the set of all points p between to endpoints. Let $I = <l,r> = \{p \mid l \leq p \leq r\}$ where I represents a time interval and $<l,r>$ is an ordered pair such that l is the left endpoint and r is the right endpoint. (Note that, by transitivity, $l \leq r$.) Also, we can define a time interval as the set of points p that occur within some time limit following a start point. Let $I = <s,\delta> = \{p \mid s \leq p \leq s + \delta\}$ where I represents a time interval as above and $<s,\delta>$ is an ordered pair such that s is the starting point and δ is the interval's duration.

Interval Based Representations

James Allen raised several questions regarding the time-point primitive and decided to assume the interval was primitive. (See reference 1 for his discussion on the point/interval issue.) He then developed a set of 13 binary relations on these intervals and a constraint propagation algorithm to compute the transitive closure of these relations through a network of intervals.

A nice feature of the interval primitive is that the interval is completely self-contained. Thus, the concern is simply with events, and specific points in time become irrelevant unless they, again, are specific events. Another nice feature is the ease in representing ambiguity. Consider the two examples in the previous section. These two situations may be represented as I_a (Equals Starts Is-Started-By) I_b and I_a (Before After) I_b.

On the other hand, a major disadvantage appears when one performs the transitive closure. Vilain and Kautz showed that the closure of intervals is reducible to the satisfiability problem which is known to be NP-complete.[2] Although Allen's algorithm for propagating time constraints executes in polynomial time, Allen took certain shortcuts to achieve this result. He avoided the combinatorial explosion by verifying consistency through only three adjacent intervals. Vilain and Kautz, on the other hand, adopt algorithms employing a point-based representation and find the algorithms to be more efficient than those operating on an interval-based representation.

REPRESENTING END POINT RELATIONS WITH INEQUALITIES

Because of the intractability of closure on interval-based networks, we will proceed from a point-based representation. In particular, we will represent a time interval by its endpoints and, for the time being, we will not concern ourselves with the problem of two intervals overlapping by a single point or the problem of representing all ambiguities.

Matrix Representation of Point Relations

We are now ready to address the question, "How do temporal events relate to one another?" First, consider two arbitrary points, p_a and p_b. There are only three ways p_a is able to relate to p_b (p_a before p_b, p_a after p_b, and p_a at the same time as p_b). Thus, given p_a rel p_b where rel $\in \{<, >, =\}$, we can define the following.

$p_a < p_b \equiv p_a$ before p_b
$p_a > p_b \equiv p_a$ after p_b
$p_a = p_b \equiv p_a$ at the same time as p_b.

Next, consider two intervals, $I_a = <l_a,r_a> = \{p \mid l_a \leq p \leq r_a\}$ and $I_b = <l_b,r_b> = \{p \mid l_b \leq p \leq r_b\}$. We can specify all possible relations between the endpoints as l_a rel_1 l_b, l_a rel_2 r_b, r_a rel_3 l_b, and r_a rel_4 r_b, where $rel_i \in \{<, >, =\}$ and $i = 1 \ldots 4$. Without considering the limitation imposed that $l_i \leq r_i$ (from our definition of an interval), with four relations and three choices for each relational operator, the total possible combinations of relations (which we will call relation signatures) is 3^4 or 81. If we impose the constraint that $l_i \leq r_i$, we find that the number of possible relation signatures reduces to 18.

BINARY RELATIONS ON TIME INTERVALS

Proceeding from the 18 relation signatures, which we will call the set of relevant relation signatures, we will divide the set into two major subsets: the set of interval relations and the set of point-interval relations. Had we proceeded from the assumption that the interval was primitive we would have found only 13 relations. These 13 relations are shown with graphical representations and relation signatures in Figure 1. Because we combined point

Relation	Symbol	Signature	Example
Equals	(A=B):		
Before	(A<B):		
Meets	(A≤B):		
Starts	(A ⊢B):		
Finishes	(A ⊣B):		
Contains	(A⊃B):		
Overlaps	(A/B):		
After	(A>B):		
Met by	(A≥B):		
Started by	(A ⊣B):		
Finished by	(A=⊣B):		
During	(A⊂B):		
Overlapped by	(A\B):		
Point Equals	(A≟B):		
Point Starts	(A⊢B):		
Point Finishes	(A⊣B):		
Point Started by	(A ⊢B):		
Point Finished by	(A⊣B):		

Figure 1. Temporal Relations

and interval constructs we were able to identify 5 additional relations that are special cases of 5 of the interval relations. These special cases arise when one or both of the intervals are time points. These signatures are also shown in Figure 1. We chose the relation labels shown in Figure 1 because they uniquely characterize the corresponding relations.

CONSTRAINT PROPAGATION ON TEMPORAL INTERVALS

Previous sections described a relational algebra on the temporal intervals. Given two intervals we have described 18 ways to specify how these intervals may relate to one other in terms of their end points. The problem discussed in this section is, given a set of intervals and relationships between a subset of these intervals, how are any two intervals related? This problem can further be divided into two subproblems. The first may be stated as follows: Given two intervals, I_i and I_j in set I such that their endpoints are known, how are I_i and I_j related? The second problem is similar. Given three intervals, I_i, I_j, and I_k in set I such that the endpoints are unknown, I_i rel I_j is known, and I_j rel I_k is known, how are I_i and I_k related? Due to space constraints, an algorithm for solving only the latter problem will be discussed.

Let us represent the relation signatures as 2x2 matrices. First, we define the addition and multiplication operators on our relational algebra in Figure 2.[2] With these operators available, we can propagate the intervals' relational constraints by multiplying the two intervals' relation matrices together (using standard matrix addition and subtraction). The matrix multiplication algorithm may be used to define a transitivity table for all pairwise combinations of interval and point-interval relations. We provide this table (Table 1) using our relation symbols and including the point-interval relations. (Note: Allen gives this table for the 12 interval relations and excludes Equals.) We can then use a constraint propagation algorithm, such as the one given in reference 1, to determine higher order relations between the intervals. Unfortunately, as mentioned earlier, this algorithm has exponential complexity which is unacceptable for large problems. In the next section, we will describe a simpler algorithm that, though incomplete, provides a polynomial time solution that provides excellent coverage.

+	<	>	=	?
<	<	0	0	<
>	0	>	0	>
=	0	0	=	=
?	<	>	=	?

x	<	>	=	?
<	<	?	<	?
>	?	>	>	?
=	<	>	=	?
?	?	?	?	?

0 = { }
? = { < > = }

Figure 2. Relational Algebra Operators

REPRESENTING TEMPORAL RELATIONS WITH PROPOSITIONAL CALCULUS

Now that we have an algebra for representing and operating on temporal relations, the next step is to map this algebra into a knowledge based inference system. Given knowledge about how interval end points relate to one another, we would like to propagate this information throughout the resulting knowledge network.

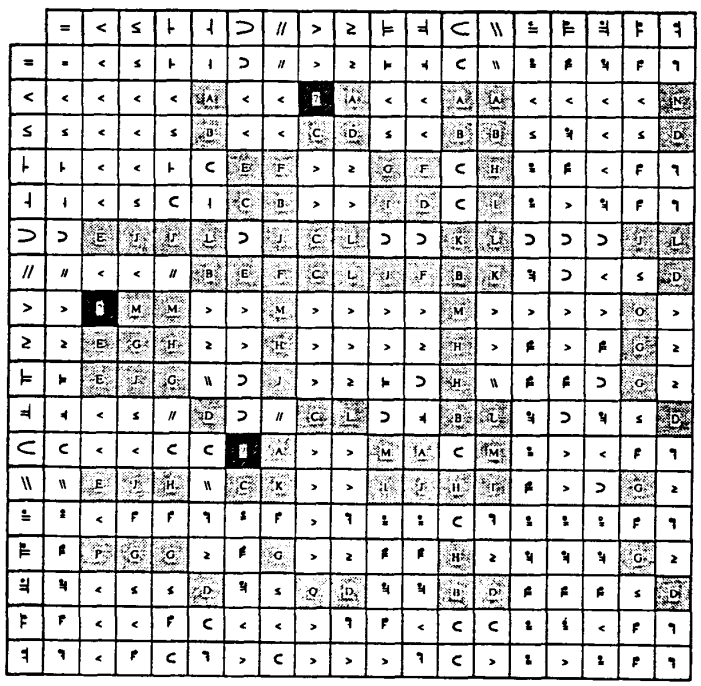
Propositional Calculus Representation of Inequality

Our relational algebra is a three value logic that is cumbersome to work with. Since digital computers are binary machines it is desirable for the sake of efficiency to map this three valued logic into a two value logic. As we will soon see this will allow for a rapid closure algorithm with respect to the end points of the temporal network. Unfortunately, mapping from a three space to a two space will result in some loss of information. Fortunately, there is a way to recover much of this lost information.

As we begin the process of mapping our three-valued logic into the two-valued logic, we consider two points in time, a and b, such that a occurs before b. We represent this, using our temporal algebra, as $(a < b)$. We also know that the inverse of $(a < b)$ is $(b > a)$; however, this still leaves us with the three-valued logic. Assume now that we use the binary relation "<" (i.e., less than) to represent the temporal relation between a and b. Then, if we wish to consider the inverse of $(a < b)$, we say b is not before a. Therefore, a may be after b, or a may occur at the same time as b. Under this specification, the inverse of $(a < b)$ is $(b \geq a)$.

Using propositional calculus, the above discussion translates into the following. We are interested in propagating a truth value through a logic network. This propagation occurs by chaining rules of inference together. These rules of inference correspond to an implication in propositional calculus. Thus, if we want to say "If A is true, then B is true," we write $(A \rightarrow B)$. At the same time, we know from propositional calculus that $(A \rightarrow B)$ is equivalent to $(\neg B \rightarrow \neg A)$. In other words, if B is false, then A is also false. Returning to our temporal system, we know if $(a < b)$, then for b to have occurred, a must also have occurred. Thus, the truth of b implies the truth of a. Conversely, if a has not occurred, b cannot have occurred. Therefore, the falsity of a implies the falsity of b.

Table 1. Transitivity Table for All 18 Relations*



A: (< ≤ // ⊂ ⊢)
B: (// ⊂ ⊢)
C: (> ≥ \\ ⊃ ⊨)
D: (= ⊣ ⊲)
E: (< ≤ // ⊃ ⊲)
F: (< ≤ //)
G: (= ⊢ ⊨)
H: (⊣ \\ ⊂)
I: (> ≥ \\ )
J: (// ⊃ ⊲)
K: (= ⊢ ⊣ ⊨ ⊲ // \\ ⊂ ⊃)
L: (\\ ⊃ ⊨)
M: (> ≥ \\ ⊂ ⊣)
N: (< ≤ // ⊂ ⊢ ≠)
O: (> ≥ \\ ⊂ ⊣ ≠)
P: (< ≤ // ⊃ ⊲ ≠)
Q: (> ≥ \\ ⊃ ⊣ ≠)

*Shading highlights ambiguities indicated by the corresponding letter. A question mark (?) indicates no information may be inferred from the corresponding relation transitivity.

This is equivalent to saying $(b \rightarrow a)$. Note also that if $(b = a)$, the above implication still holds. In fact, equality would translate into equivalence, i.e., $(a \equiv b)$. Therefore, we may use this mapping to determine propositional rules for our 18 temporal relations.

Mapping Temporal Relations

As we begin to determine the propositional rules corresponding to each temporal relation, we recall the relation signatures defining the relations between each of the end points. First we will state a couple of obvious rules based on our definition of a temporal interval. Then we will delineate the propositions that arise from the relation signatures.

Recall from our definition that we declared the left endpoint of an interval either to be equal to the right endpoint (a point interval) or to be before the right endpoint (a non-point interval). For point intervals, we have $(l_i = r_i)$ which translates in propositional calculus to $(l_i \equiv r_i)$ or, using implication, $(l_i \rightarrow r_i) \wedge (r_i \rightarrow l_i)$. For non-point intervals, $(l_i < r_i)$; therefore, $(r_i \rightarrow l_i)$. Another obvious observation is that any given point equals itself; therefore, $(l_i \rightarrow l_i)$ and $(r_i \rightarrow r_i)$. The reason for explicitly stating this tautology will become apparent below. The rules corresponding to mapping each of the relation signatures are given in Table 2.

Table 2. Propositional Rules for Temporal Relations

Relation	Mapping Rules
A = B	(la -> lb) ^ (lb -> la) ^ (rb -> ra) ^ (ra -> lb) ^ (ra -> rb) ^ (rb -> ra)
A $\stackrel{s}{=}$ B	(la -> lb) ^ (lb -> la) ^ (la -> rb) ^ (rb -> la) ^ (ra -> lb) ^ (lb -> ra) ^ (ra -> rb) ^ (rb -> ra)
A < B	(lb -> la) ^ (lb -> ra) ^ (rb -> la) ^ (rb -> ra)
A > B	(la -> lb) ^ (ra -> lb) ^ (la -> rb) ^ (ra -> rb)
A ≤ B	(lb -> la) ^ (lb -> ra) ^ (ra -> lb) ^ (rb -> la) ^ (rb -> ra)
A ≥ B	(la -> lb) ^ (la -> rb) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb)
A |- B	(la -> lb) ^ (lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (rb -> ra)
A |= B	(la -> lb) ^ (lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb)
A -| B	(la -> lb) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb) ^ (rb -> ra)
A =| B	(lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb) ^ (rb -> ra)
A ⊃ B	(lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb)
A ⊂ B	(la -> lb) ^ (rb -> la) ^ (ra -> lb) ^ (rb -> ra)
A // B	(lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (rb -> ra)
A \\ B	(la -> lb) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb)
A $\stackrel{f}{=}$ B	(la -> lb) ^ (lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (lb -> ra) ^ (rb -> ra)
A $\stackrel{f}{\leq}$ B	(la -> lb) ^ (lb -> la) ^ (la -> rb) ^ (rb -> la) ^ (ra -> lb) ^ (rb -> ra)
A $\stackrel{f}{\geq}$ B	(la -> lb) ^ (la -> rb) ^ (rb -> la) ^ (ra -> lb) ^ (ra -> rb) ^ (rb -> ra)
A $\stackrel{d}{=}$ B	(lb -> la) ^ (rb -> la) ^ (ra -> lb) ^ (lb -> ra) ^ (ra -> rb) ^ (rb -> ra)

Table 3. Bit Signatures for Temporal Relations

	la	ra	lb	rb
A = B	1	0	1	0
	1	1	1	1
= <	1	0	1	0
> =	1	1	1	1
A > B	1	0	1	1
	1	1	1	1
> >	0	0	1	0
> >	0	0	1	1
A |- B	1	0	1	0
	1	1	1	0
= <	1	0	1	0
> <	1	1	1	1
A -| B	1	0	0	0
	1	1	1	1
< <	1	0	1	0
> =	1	1	1	1
A // B	1	0	0	0
	1	1	1	0
< <	1	0	1	0
> <	1	1	1	1
A $\stackrel{f}{=}$ B	1	0	1	1
	1	1	1	1
= =	1	0	1	1
> >	1	0	1	1

	la	ra	lb	rb
A $\stackrel{s}{=}$ B	1	1	1	1
	1	1	1	1
= =	1	1	1	1
= =	1	1	1	1
A ≤ B	1	0	0	0
	1	1	1	0
< <	1	1	1	0
= <	1	1	1	1
A |= B	1	0	1	0
	1	1	1	1
= <	1	0	1	0
> >	1	0	1	1
A ⊃ B	1	0	0	0
	1	1	1	1
< <	1	0	1	0
> >	1	0	1	1
A \\ B	1	0	1	0
	1	1	1	1
> <	0	0	1	0
> >	1	0	1	1
A $\stackrel{f}{\geq}$ B	1	1	1	1
	1	1	1	1
> =	0	0	1	0
> =	1	1	1	1

	la	ra	lb	rb
A < B	1	0	0	0
	1	1	0	0
< <	1	1	1	0
< <	1	1	1	1
A ≥ B	1	0	1	1
	1	1	1	1
> =	0	0	1	0
> >	1	0	1	1
A | B	1	0	1	0
	1	1	1	1
> <	0	0	1	0
> =	1	1	1	1
A ⊂ B	1	0	1	0
	1	1	1	0
> <	0	0	1	0
> <	1	1	1	1
A $\stackrel{f}{=}$ B	1	1	1	0
	1	1	1	0
= <	1	1	1	0
= <	1	1	1	1
A $\stackrel{d}{=}$ B	1	0	0	0
	1	1	1	1
< <	1	1	1	1
= =	1	1	1	1

	la	ra	lb	rb
	la			
	ra			
	lb			
	rb			

Note: If Matrix(i,j) = 1, then row(i) -> column(j)

Bit Matrix Signatures

Having specified the propositional rules corresponding to each of 18 relation matrices, we will now represent these logical relations in the machine. These logical relations may be represented as a two dimensional binary matrix. For any given cell in the binary matrix, a zero indicates it is not known if the point specified by the row implies the point specified by the column. A one, on the other hand, indicates an implication is known. For example, a one in row 2, column 3, indicates $(r_a \rightarrow l_b)$. In other words, $(l_b < r_a)$. If a one is in row 2, column 3 and row 3, column 2, then we can say $(l_b = r_a)$. This is because $(l_b \rightarrow r_a) \wedge (r_a \rightarrow l_b)$, i.e., $(l_b \equiv r_a)$. All the bit matrix signatures for the 18 temporal relations are shown in Table 3.

CONSTRAINT PROPAGATION OF BINARY SIGNATURES

In order to represent a temporal network we will now specify a data structure for representing temporal intervals in our system. This data structure consists of a $2n$ x $2n$ binary matrix where n is the number of intervals in the system. The relations between the intervals are then entered into the matrix according to the relation signatures given

in Table 3. Once the matrix is set up we are ready to propagate the temporal constraints through the knowledge network.

Transitive Closure of Bit Matrices

We begin by describing a basic algorithm for the transitive closure of a graph. We use this algorithm to perform the constraint propagation. The details of the algorithm are given in reference 3. Recall, however, that by representing the temporal relations in a binary form, we have lost information. The next section will describe an approach to recovering some of the lost information.

Signature Analysis and Ambiguity

Following closure, we can determine the relationships between any pair of intervals by examining their corresponding submatrices and comparing the resulting signature with the table of primitive signatures. If an exact match is found, then the relationship has been identified. In the event the relationship is ambiguous (i.e., more than one relation is consistent for a pair of intervals), then an exact match will not be found. In order to resolve the ambiguity, the following procedure may be followed:

1. Examine all entries in the matrix above the diagonal and compare them with their corresponding entries below the diagonal (i.e., M(a, b) compared to M(b, a)).

2. If M(a, b) and M(b, a) = 0, set both M(a, b) and M(b, a) to 1 in the submatrix (not the original matrix).

3. Perform a logical "AND" between the modified submatrix and each signature matrix.

4. If the resulting matrix is equivalent to the signature matrix, then the relation corresponding to the signature matrix is a member of the ambiguity group.

THE APPLICATION OF TEMPORAL CONSTRAINT PROPAGATION TO HARDWARE TESTABILITY AND FAULT DIAGNOSIS

Timing in hardware systems has long presented problems in testability and fault-isolation analysis. We designed the System Testability and Maintenance Program (STAMP) to address the question of system testability using an information flow model approach.[4] In effect, modeling may proceed from an analysis of the functionality of a system and the corresponding failure modes of that system. One example of a hardware construct that is highly time sensitive is the bus structure. Bus structures control the transfer of information between points in a system and rely upon proper timing of the transfer. In addition, other portions of the system rely on the timing of the bus to ensure information is arriving at the time needed.[5]

As a result of this work in temporal reasoning, we devised several temporal dependency paradigms that take advantage of endpoint relations. (A detailed discussion of these paradigms is beyond the scope of this paper and will not be discussed here.) Tests defined for various endpoint relations resolve interval relation ambiguity, so the ambiguity propagation drawback is not seen as a problem in this area of analysis.

These temporal paradigms were used in a sample system developed for a reconfiguration expert system. The system consisted of five major functions, two of which had to operate concurrently. One of the two functions contained a structure similar to a bus which further complicated the model by creating a feedback loop. The temporal paradigms succeeded in enabling isolation to a timing problem and broke up the feedback loop as well. Once fault isolation was complete; the reconfiguration expert proceeded to locate available system components to compensate for the fault and reconfigured the system using those components to make the system functional again.

SUMMARY

This paper presented a summary of work done in developing an algebra of relations for temporal reasoning. It then proceeded to extend the work using an interval based approach, but incorporating point intervals in the model. Finally a propositional calculus representation of the temporal relations was derived and combined with the transitive closure algorithm that operates on a bit matrix. The result was an efficient and relatively simple approach to modeling relations between temporal intervals and propagating the constraints imposed by these relations through the knowledge base.

REFERENCES

1. James Allen, "Maintaining Knowledge About Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, November 1983.

2. Mark Vilain and Henry Kautz, "Constraint Propagation Algorithms for Temporal Reasoning," *Proceedings of AAAI-86*, Philadelphia, PA, August 11-15, 1986.

3. John W. Sheppard and William R. Simpson, "A Mathematical Model for Integrated Diagnostics," *IEEE Design and Test of Computers*, Vol. 8, No. 4, December 1991, pp. 25-38.

4. William R. Simpson and John W. Sheppard, "System Complexity and Integrated Diagnostics," *IEEE Design and Test of Computers*, Vol. 8, No. 3, September 1991, pp. 16-30.

5. Frank Johnson and Eugene Esker, "Bus Structure Modeling Methods," *STAMP Technical Note*, No. 412, ARINC Research Corporation, January 1988.