UNCERTAINTY COMPUTATIONS IN MODEL-BASED DIAGNOSTICS

John W. Sheppard and William R. Simpson
ARINC Research Corporation
2551 Riva Road
Annapolis, Maryland 21401

ABSTRACT

A number of model-based diagnostic reasoning systems have been developed that are based on an inferential approach to fault isolation where the system model provides the diagnostic rules of inference. Field applications of reasoning systems have revealed that model-based techniques are preferable to traditional diagnosis methods, and in some cases these systems are performing at or above the level of the system experts. There exists a class of problems for which the inferential process is inadequate and must be supplemented with forms of reasoning under uncertainty. This class of problems is typified by uncertain test outcomes, inadequate technician skill levels to test and interpret the results, and even uncertain elements in the model. These difficulties are amplified in that, even in a nominally well-behaved problem, small portions involving these shortcomings dictate that the entire problem be treated as a reasoning-under-uncertainty or evidence-gathering problem.

A number of paradigms exist for reasoning with knowledge bases that are incomplete, inaccurate, or conflicting. In this paper, we describe an approach to reasoning under uncertainty that combines several basic computational procedures of artificial intelligence with information modeling approaches used in the System Testability and Maintenance Program (STAMP®). These techniques have been incorporated into the Portable Interactive Troubleshooter (POINTER™) to provide a model-based diagnostic reasoning system that handles inaccurate, incomplete, and conflicting data.

We briefly describe the computational methods used in uncertainty data gathering (modified fuzzy logic), evidence accumulation (modified Dempster-Shafer), and termination of the evidence-gathering process (back-propagation neural network) as they are implemented in the POINTER system.

INTRODUCTION

The 1980s saw a revolution in the approach to field maintenance for complex systems. The testability and diagnosis techniques are now being placed deeper and deeper into the system design itself. Even with the increased emphasis on testability and diagnosis, there still exist shortcomings in the field maintenance process. ARINC developed two tools to assist in designing testable systems and in streamlining the maintenance process—STAMP and POINTER. STAMP is a model-based reasoning system that can conduct testability analyses and develop fault-isolation strategies to improve system maintenance. The information modeling approach incorporated by STAMP permits analysis of a wide range of systems, including digital, analog, digital/analog hybrid, electrohydraulic, and electromechanical. STAMP has been used to analyze systems in various stages of the acquisition process (e.g., preliminary design, prototype, redesign, and operational). Several of these applications have included analyses of built-in test (BIT), and some have used built-in test equipment (BITE) and other forms of automatic and semiautomatic test equipment. In addition, the particular levels of analysis have varied from macro (full system) to micro (piece-part), with several levels in between—for example, line and shop replaceable units (LRUs and SRUs) or weapon and shop replaceable assemblies (WRAs and SRAs).

For many systems analyzed using STAMP, significant improvements have been achieved, and for some systems, order-of-magnitude improvements have been achieved [1, 2]. The software is mature and has been applied to more than 200 systems. It is currently being used by ARINC in several applications. POINTER, which was derived from STAMP, uses the system model generated in STAMP for its knowledge base. With the system model, POINTER interactively presents test material and guides the maintenance process through diagnosis and repair. At each step, POINTER examines the current system state to determine the next best test to perform. The basic processes of modeling and knowledge-base development, as well as the testability and fault-isolation output, are described in depth in the literature [3-5] and are not discussed in this paper.

The process of diagnosing complex systems requires determining the tests available, how to perform the tests, the appropriate order for sequencing tests, and the conclusions to be drawn from the test outcomes. In many diagnostic systems, test outcomes are assumed to be 100% certain, and diagnosis proceeds through a partitioning of the answer set into feasible and infeasible conclusions. Frequently, this approach does not adequately solve the problem because of uncertainty in the testing process.

In the artificial intelligence community, problems in which information available for reasoning is incomplete or uncertain are addressed using techniques referred to as "reasoning under uncertainty," and several approaches exist.

CH2941-3/91/0000-0233 $1.00

Some of the basic formulations of the problem use certainty factors, Bayesian probability, and weighted causal networks. In addition, there are various forms of logic that allow some aspects of uncertainty to be considered, such as predicate calculus, multivalued logics, modal logics, nonmonotonic logics, and intuitional logics. As part of its 1990 internal research and development program, the ARINC Advanced Research and Development Group incorporated an approach to "reasoning under uncertainty" into the POINTER system. The approach is based on a modified form of Dempster-Shafer evidential reasoning, which we describe in this paper [6-10]. Three areas of research in particular have influenced our approach to uncertainty: fuzzy logic [11-12], the theory of evidence [6-7], and neural networks [10, 13].

FUZZY LOGIC

In traditional set theory, some element either does or does not belong to a given set. More formally, given a set S, a membership function μ, and some element x, we say μ: domain(x) $\rightarrow$ {0,1} where

$$\mu(x) = \begin{cases} 1; x \in S \\ 0; x \notin S \end{cases}.$$

Unfortunately, traditional set theory is usually too limiting in its application to real-world problems. In the area of fault diagnosis, we traditionally begin with a set theoretic approach in which we attempt to partition the conclusions into two sets: feasible and infeasible. We also tend to partition the tests into two sets: failed and passed. But frequently, the division between these pairs of sets is not clear. We therefore need the notion of a "fuzzy set." In fuzzy set theory, we replace the traditional set theoretic membership function with one where we say μ: domain(x) $\rightarrow$ [0,1]. Thus, $0 \leq \mu(x) \leq 1$, and a fuzzy set describes a class that allows the possibility of partial membership in a set.

Fuzzy logic proceeds to develop operations to be performed on fuzzy sets analogous to the operations of traditional set theory. For example, the union and intersection operations are defined as follows:

Let S_1 and S_2 be two fuzzy sets. Then

$$\mu_{S_1 \cup S_2} = \max(\mu_{S_1}, \mu_{S_2}), \text{ and}$$

$$\mu_{S_1 \cap S_2} = \min(\mu_{S_1}, \mu_{S_2}).$$

These fuzzy operators provide a means for reasoning under uncertainty with fuzzy sets. We decided to use these concepts at the user interface and to apply other techniques for actually reasoning with the fuzzy values provided by the user.

The first step in incorporating uncertainty calculations involved determining an approach for confidence values to be assigned to a test outcome. Typically, systems incorporating uncertainty draw on the user's familiarity with probability theory and ask for a confidence value such that $0 \leq Confidence \leq 1$. For our application domain (fault diagnosis), we believed that this requirement may be unreasonable for the average user. Therefore, we developed an alternative approach in which we defined a mapping from a discrete set of English qualifiers to the interval mentioned above, i.e., μ: Q $\rightarrow$ [0,1], where μ is the mapping and Q is the set of qualifiers.

The actual implementation allows up to five qualifiers to be associated with a failed test and up to five qualifiers to be associated with a passed test. In addition, a base confidence value may be assigned to each test outcome, defining an upper bound on the value of the certainty that the test outcome will ever have. Thus,

$0 \leq \mu_p(x) \leq pass\_base$, and

$0 \leq \mu_f(x) \leq fail\_base$,

where *pass_base* is the base confidence for a passed test, and *fail_base* is the base confidence for a failed test. The qualifiers are specified in decreasing order of confidence (e.g., <Certain, Somewhat Certain, Marginal, Somewhat Uncertain, Uncertain>). This may be represented as an array of qualifiers:

$Q = \langle q_1, ..., q_n \rangle$.

The corresponding confidence value associated with a qualifier is then computed as

$$Confidence = \frac{\left(base - \frac{1}{2}\right)(n - i)}{n - 1} + \frac{1}{2},$$

where *Confidence* is the confidence value, *base* is the appropriate base confidence, n is the number of qualifiers in Q, and i is the index of the selected qualifier. This form of user interface has proved to be more acceptable than those forms in which the user provides the outcome confidences.

DEMPSTER-SHAFER EVIDENTIAL REASONING

Having a method for determining the user-supplied confidence associated with a particular test outcome, we must use that confidence in the inference process to determine what has failed. Our approach consists of five steps:

1. Select a test to perform.
2. Obtain the test outcome and confidence.

3. Draw inferences based on the test outcome.
4. Derive hypotheses.
5. Determine if sufficient evidence has been gathered to draw a conclusion.

Selecting a Test

The process of selecting a test to perform is conducted in two phases, which follow an entropy-based approach and an evidence-based approach, respectively. During the first phase, tests are selected under the assumption that tests provide perfect information, even though they may not. To choose a test, we use information theory to determine the amount of information to be gained by each test, independent of test outcome. The amount of information is further modified by a cost parameter. From information theory [14], we know that the expected value of information gained by an information source (called the information entropy) can be computed from the following equation.

$$H(\mathbf{O}) = \sum_{i=1}^{m} -\Pr(O_i) \log_2 \Pr(O_i),$$

where H is the information entropy, **O** is the set of possible outcomes, O_i is the ith member of set **O**, and m is the number of possible outcomes in **O**. In the case where there are only two outcomes (pass or fail), this reduces to

$$H(t) = -\Pr(t\text{=pass}) \log_2 \Pr(t\text{=pass}) - \Pr(t\text{=fail}) \log_2 \Pr(t\text{=fail}).$$

If we assume that the probability that a test will pass or fail is 0.5, then we find that the problem of computing information entropy is reduced to the following:

$$H(t) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 0.5 + 0.5 = 1.$$

Using a dependency-based formulation, this means that we can compute the information gained by counting test inferences. We can ensure that we obtain the most robust information by choosing the test with the highest minimum value, given by

$$I(t) = \max_j \left\{ \min \left\{ \sum_{i=1}^{n} \delta_{ij}^{p} , \sum_{i=1}^{n} \delta_{ij}^{f} \right\} \right\},$$

where

$$\delta_{ij}^{p} = \begin{cases} 1 \text{ ; if } t_j \text{ depends on } t_i \\ 0 \text{ ; otherwise} \end{cases}, \text{ and}$$

$$\delta_{ij}^{f} = \begin{cases} 1 \text{ ; if } t_i \text{ depends on } t_j \\ 0 \text{ ; otherwise} \end{cases}.$$

Note that we assume a test depends on itself.

For many applications, this approach to selecting a test is too limited. In particular, we are interested in optimizing the diagnostic process according to some cost criterion (or set of cost criteria). Our approach is to combine whatever cost criteria are specified into a single test weight that is then applied to the information count of the corresponding test. Test costs in general do not differ on the basis of expected outcome; however, one cost weight may be the test base confidence, which may differ for pass and fail outcomes. This fact results in a minor modification to the preceding equation. Using a cost weight, the test choice algorithm becomes choosing a test such that

$$I(t) = \max_j \left\{ \min \left\{ w_j^p \sum_{i=1}^{n} \delta_{ij}^{p} , w_j^f \sum_{i=1}^{n} \delta_{ij}^{f} \right\} \right\},$$

where w_j^p is the cost applied to test j being passed, and w_j^f is the cost applied to test j being failed.

When a value (i.e., pass, fail, unavailable, unneeded) has been determined for each test in the model, then the test-choice process passes into the second phase. At this point, a fault hypothesis, **H**, is determined for the system based on the testing performed (see Deriving Hypotheses), and this hypothesis is used as the basis for the test choice. This process considers four alternatives:

1. Splitting multiple hypotheses
2. Increasing support for a single hypothesis
3. Decreasing support for a single hypothesis
4. Arbitrarily selecting a test

Any one of these alternatives can fail to select a test for various reasons. The most important reason, however, is that we impose a limit on the number of times a test can be chosen. Once that limit is exceeded, we remove the test from consideration and perform the selection algorithm on the remaining tests. If no tests are available to meet the requirements of a particular option, then the next option is considered.

Splitting Multiple Hypotheses. When two or more hypotheses are being considered, POINTER attempts to choose a test that will support some of the hypotheses and deny other hypotheses. The approach taken is similar to the entropy-based approach described for the first phase. The primary difference is that dependency relationships between tests and conclusions are considered rather than the relationships between tests and tests. A test is chosen such that

$$E(t) = \max_j \left\{ \min \left\{ \sum_{i=1}^{n} \delta_{ij}^{1} , \sum_{i=1}^{n} \delta_{ij}^{2} \right\} \right\},$$

where $E(t)$ is the expected evidence to be gained by performing test t, and, given that h_i is the ith member of the hypothesis set, **H**,

$$\delta_{ij}^{1} = \begin{cases} 1 \text{ ; if } t_j \text{ depends on } h_i \\ 0 \text{ ; otherwise} \end{cases}, \text{ and}$$

$$\delta_{ij}^{2} = \begin{cases} 0 \text{ ; if } t_j \text{ depends on } h_i \\ 1 \text{ ; otherwise} \end{cases}.$$

Increasing Support for a Single Hypothesis. When only one hypothesis exists (or multiple hypotheses are in an ambiguity group) and the first alternative fails to select a test, then POINTER attempts to identify a test that will increase the support for the current hypothesis. For this alternative, POINTER selects a test such that

$$E(t) = \max_j \left\{ \sum_{i=1}^{n} \delta_{ij}^{s} \right\},$$

where

$$\delta_{ij}^{s} = \begin{cases} 1 \text{ ; if } t_j \text{ depends on } c_i \text{ and } \mathbf{H} \\ 0 \text{ ; otherwise} \end{cases}.$$

Decreasing Support for a Single Hypothesis. If POINTER does not choose a test from either of the first two alternatives, then it is possible that the current hypothesis is a poor one. On the other hand, it is possible to improve upon the current hypothesis if we select a test that, should it fail, decreases support for the hypothesis. So we choose a test that considers as many conclusions in the system as possible *other than* the current hypothesis. In other words, we choose a test such that

$$E(t) = \max_j \left\{ \sum_{i=1}^{n} \delta_{ij}^{d} \right\},$$

where

$$\delta_{ij}^{d} = \begin{cases} 1 \text{ ; if } t_j \text{ depends on } c_i \text{ but not } \mathbf{H} \\ 0 \text{ ; otherwise} \end{cases}.$$

Arbitrary Test Selection. Finally, if none of the procedures discussed results in a test being chosen, POINTER picks the last test in the list of available tests to be performed. This results in some test being chosen so that additional evidence can be gathered. If this option fails, then there are no tests left to be evaluated, and the user is offered the option of choosing a test or terminating.

Obtaining a Test Outcome and Confidence

At this point, POINTER presents the test for evaluation, and the test is then performed. In automatic testing, the test confidence may be provided by the program run by the ATE, or it may be defaulted to the base confidence. If field maintenance is being conducted, then the technician is responsible for performing the test and providing the result. The method for specifying confidence in the outcome for manual testing is discussed under Fuzzy Logic.

Drawing Inferences Based on the Test Outcome

In the section on fuzzy logic, we discussed the notion of a test confidence. The test confidence is intended to be a measure of the tester's confidence that the outcome obtained is correct. For example, a 0.9 confidence in a test outcome means that we believe there is a 90% chance that the test outcome is correct. Because we associate confidences with both pass and fail outcomes, it may be possible that we will have a 90% confidence in a passed test but only an 80% confidence in a failed test. Thus, the two confidences are not necessarily complementary or equivalent. A 50% confidence in any test outcome is considered to be a totally uncertain outcome.

These confidence values can be used to perform a statistical inference procedure on the set of conclusions in the model. The Dempster-Shafer approach to reasoning with uncertain data has its roots in Bayesian inference. If we assume we know the probability that each component may fail ($\Pr[c_i]$, which is also the probability that each conclusion will be drawn) and the probability that some symptom will occur with the corresponding failure ($\Pr[symptom \mid c_i]$), then we can apply Baye's rule to determine the probability that the component is indeed the failure, given the symptoms, as follows:

$$\Pr[c_i|s] = \frac{\Pr[c_i]\ \Pr[s|c_i]}{\sum_k \Pr[c_k]\ \Pr[s|c_k]},$$

where c_i is the ith possible conclusion in the set of conclusions and s is the combination of all evidence gathered so far (i.e., the set of symptoms known).

Shafer added the concept of uncertainty to the Bayesian approach, which resulted in applying two measures to a conclusion—support and plausibility [6]. The evidence generated by a test is assigned to each related conclusion as either support evidence or denial evidence. A test supports a conclusion when the test outcome is consistent with the component that is concluded to be the fault. A test denies a conclusion when the test outcome is not consistent with the component that is concluded to be the fault. Support functions allocate evidence gathered by a test measurement to a set of one or more conclusions. Each test has two support functions, one for each type of evidence (support and denial). These functions identify the conclusions related to a given test and allocate the evidence provided by that test. The values for *Support* and *Plausibility* define a credibility interval for belief in some conclusion and have the property $Support_i \le \Pr[c_i] \le Plausibility_i$. Further, both support and plausibility are limited to the closed interval

[0,1]. Support represents the degree to which the evidence supports the conclusion, and plausibility represents the degree to which the evidence fails to refute the conclusion. It should also be clear that, because $Support_i \leq Plausibility_i$, $Support_i + (1 - Plausibility_i) \leq 1$. (Note: *Plausibility* is simply 1 - *Denial*, where *Denial* is given below.)

In determining the support and plausibility for a conclusion, Shafer began with the confidence value. In determining the support for a conclusion, we find that a piece of information may support multiple conclusions; thus the evidence in support of these conclusions is distributed over the set of supported conclusions. For our formulation, we uniformly distribute the support that yields the following:

$$Support_i = \frac{Confidence}{|C_s|},$$

where C_s is the set of conclusions supported by the evidence. In addition, any evidence that denies a conclusion may deny multiple conclusions. However, rather than distribute the denial uniformly, we apply the full weight of denial on every conclusion in the set. Thus,

$$Denial_i = Confidence.$$

Note that, to attribute support or denial because of some confidence in the connection between an information source and conclusion, we may simply apply a weight to these two expressions, yielding the following equations:

$$Support_i = w_i^s \frac{Confidence}{|C_s|}, \text{ and}$$

$$Denial_i = w_i^d \, Confidence,$$

where w_i^s is a weight indicating the amount of confidence we have that a piece of evidence supports c_i, and w_i^d is a weight indicating the amount of confidence we have that a piece of evidence denies c_i.

Unfortunately, these calculations are not sufficient for drawing inferences with multiple pieces of evidence. This fact led to the development of the rule of combinations by Dempster [7]. If we assume we have a probability mass for a conclusion set associated with a single source of evidence, $\mu_1(C_1)$, and a probability mass for another conclusion set associated with a second single source of evidence, $\mu_2(C_2)$, then we can compute the probability mass associated with the intersections of these two sets based on the two sources of evidence as follows:

$$\mu_3(C_1 \cap C_2) = \frac{\sum_{C_1 \cap C_2} \mu_1(C_1)\ \mu_2(C_2)}{1 - \sum_{C_1 \cap C_2} \mu_1(C_1)\ \mu_2(C_2)}.$$

As tests are performed, Dempster's rule is applied iteratively rather than all at once, allowing evidence to be accumulated serially, which in turn allows the diagnostic process to be optimized. The actual implementation of Dempster's rule is performed in three steps. First, the intersections of the two sets, C_1 and C_2, are found. Next, the two measures of evidence corresponding to these two sets are combined, using a modification of Dempster's rule. Finally, the combined evidence is adjusted for each proposition in the intersection to reflect the difference of the evidence from the two original sets. Let K represent a normalizing constant for Dempster's rule, given simply as

$$K = \sum_{C_1 \cap C_2} \mu_1(C_1)\ \mu_2(C_2).$$

Then the first step in computing combined support is as follows:

$$TSupport_i = TSupport_i\ (Support_i + Uncertainty) + Support_i\ (1 - Confidence),$$

where $TSupport_i$ is the accumulated support for c_i for this test, and *Uncertainty* is a measure of uncertainty accumulated so far. *Uncertainty* is then updated with the following:

$$Uncertainty = Uncertainty \frac{1 - Confidence}{1 - K}.$$

Also, in order to reduce the influence of conclusions with very low support, we gradually reduce the support if $TSupport_i$ is less than $|C|^{-1}$. This reduction is performed by reducing the combined support by 25%.

A given fault-isolation problem is initialized to have 100% uncertainty with each answer, given no support and no denial (i.e., 0% support and 100% plausibility). This is one of many approaches that could have been used to begin diagnosis. One possible alternative is to initialize the support values based on the probability of failure given by failure rate data. The problem with this approach is that these probabilities may unduly weight the evidence-gathering process such that inconsistent conclusions are still considered as hypotheses.

The Dempster-Shafer technique is not without flaws. One of the greatest flaws in the technique lies in the way it records total uncertainty (computed as above). If any test is performed that provides any evidence in support of some conclusion, then uncertainty is reduced—even in the event of a conflict with known information! Ultimately, uncertainty disappears altogether. This is not satisfactory, so we included a new conclusion in our approach: the unanticipated result. The unanticipated result is never denied (i.e., it always has a plausibility of 1.0), and it is supported only when a test outcome is inconsistent (i.e., conflicts) with previous test outcomes. Thus, uncertainty

becomes a combination of Dempster-Shafer uncertainty and the support for the unanticipated result.

To compute the support for the unanticipated result, we need to choose a hypothesis. We discuss the process of selecting a hypothesis in the next section. Given a hypothesis $\mathbf{H} \subseteq \mathbf{C}^+$ (a non-empty set of conclusions), a conflict results when a test outcome denies *all* members of **H**. The support for the unanticipated result ($Support_u$) is computed in two steps. First, the unanticipated result is apportioned over the number of tests executed so that the overall conflict is reduced by the number of tests. We gradually reduce $Support_u$ as follows:

$$Support_u = Support_u \frac{T - 1}{T},$$

where T is the number of tests performed so far. If a conflict has resulted (from a test denying all conclusions in **H**), $Support_u$ is updated to reflect this.

$$Support_u = Support_u + \frac{Conflicts\ K\ Confidence}{T},$$

where *Conflicts* is the number of times a conflict has occurred.

The final values for support and plausibility are calculated as follows:

$$Support_i = \frac{Support_i\ (1 - Support_u)}{\sum_{\forall c \in \mathbf{C}} Support_c},$$

$$Denial_i = Denial_i + TDenial_i, \text{ and}$$

$$Plausibility_i = 1 - \frac{Denial_i}{T},$$

where $Denial_i$ is the total denial for $c_i \in \mathbf{C}$, and $TDenial_i$ is the total denial accumulated from this test for c_i.

Derive the Hypotheses

A key element in computing the evidential statistics is the determination of a hypothesis. In order for tests to be selected, we must consider the nature of the hypothesis to determine if the set contains one or more elements. We must also consider the effects of performing tests on the change in confidence in the hypothesis. The new conclusion—the unanticipated result—addresses the problem of disappearing uncertainty and conflict management but receives support only in the event a test denied all members of a hypothesis set, **H**. Therefore, the problem now is to construct this hypothesis set.

To choose conclusions for **H**, we first compute an approximation for the Bayesian probability associated with each conclusion. This result will also be used later in determining if sufficient evidence has been gathered to terminate fault isolation. Thus, we are defining a mapping, ρ:<*Support*, *Plausibility*> → [0,1]:

$$\rho_i = \frac{1}{2}(Plausibility_i - Support_i)(1 - Plausibility_i\ Support_i) + Support_i.$$

Given this mapping, the first conclusion to be included in **H** is simply the conclusion with the maximum value for ρ. Let

$$\rho_h = \max_i \{\rho_i\}.$$

Then we can define a threshold, θ, for determining if additional conclusions should be included in **H**.

$$\theta = \begin{cases} 0.5\ \rho_h & ;\ \rho_h \le 0.1 \\ \rho_h\ (0.4 + \rho_h) & ;\ 0.1 < \rho_h < 0.5 \\ 0.9\ \rho_h & ;\ \rho_h \ge 0.5 \end{cases}.$$

Any conclusion, c_i, such that $\rho_i \ge \theta$, results in c_i being included in **H**.

Determine if Sufficient Evidence Has Been Gathered to Draw a Conclusion

The final step in the uncertainty approach incorporated into POINTER involves examining the confidences in the conclusions of the model and determining if a conclusion can be drawn. We found that the termination problem [10] was actually a pattern-recognition problem. Therefore, this step was based on a neural network approach, which is not considered a part of the Dempster-Shafer calculations.

NEURAL NETWORKS

Research in the field of neural networks has been ongoing since the early 1950s. Much of the early work attempted to combine research in neurobiology, psychology, mathematics, and computer science to model cognitive processes in the brain. The resulting models were structures that were highly parallel and highly connected. These "artificial neural networks" consist of many simple processing elements. Each element has a set of inputs that can be represented as a vector (or an array), **X**, of values. Generally, each element also has a single output. The connections between the elements in the network have "strengths" associated with them, represented in the form of a vector, **W**, of weights. Let $x_i \in \mathbf{X}$ be the ith input to the current processing element. Let $w_j \in \mathbf{W}$ be the jth connection weight between input x_i and the processing

element. Then the output of the element is the weighted sum of the inputs. This corresponds to the inner product of the two vectors, **X** and **W**. Thus,

$$y = \sum_{i=0}^{n-1} w_i x_i.$$

To control the activity of these neural processing elements as signals pass through the network, certain functions have been applied to the outputs of the elements. As a result, the output of an element is determined by

$$\Omega = f(y) = f\left(\sum_{i=0}^{n-1} w_i x_i\right),$$

where Ω is the activation function of the element. These functions are referred to as activation functions because they regulate the activity of the neurons (elements) in the network. The particular activation function we used is the logistic function because is simulates a threshold function and is differentiable:

$$f(y) = \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-\sum_i w_i x_i}}.$$

We selected the back-propagation neural network to solve the termination problem. The back-propagation network is called a mapping network because it solves the following type of problem. Assume we have a function, f, that maps a set of inputs, **I**, into a set of targets, **T** (i.e., f: **I** → **T**). Assume also that it is impossible to present all instances of **I** for training (perhaps because **I** is an infinite set). We want to learn f, using a subset of **I**, which can then be used to generalize to all instances of **I**.

As a practical approach to the problem, we can choose $\mathbf{I}_s \subseteq \mathbf{I}$ and the corresponding $\mathbf{T}_s \subseteq \mathbf{T}$. We will use these two sets to learn the map f_s: $\mathbf{I}_s \rightarrow \mathbf{T}_s$. We then want to present some $i \notin \mathbf{I}_s$; $i \in \mathbf{I}$, to f_s as follows:

f_s: $i \rightarrow \mathbf{T}$.

This corresponds to the generalization described above.

The back-propagation neural network solves this problem in the following way. A layered network is constructed in which the first layer receives the input data, and the last layer provides the output of the network. Zero or more "hidden" layers are positioned between the input and output layers. All of the nodes at a given layer are connected to every node at the next higher layer. Associated with each of these connections is a weight that is initially set to random values. The process of training the network involves modifying the weights so that the inputs are transformed to the outputs as the signal is propagated.

The term "back propagation" comes from the learning algorithm. The first time an input is passed through a network, the output has a very low probability of being correct. Back propagation involves computing an error value corresponding to how much the actual output differs from the expected output. The weights of the network are then modified gradually, following the error surface defined by the weight space down the gradient of the surface to a local minimum. The idea is to reduce the error between the actual and expected outputs, and when the network settles, it should contain a solution to the mapping problem.

The termination problem in POINTER was addressed with a three-step solution. The first step uses the back-propagation neural network we previously described [10]. This network was trained with data analyzed by 15 testability "experts" and validated with an additional data set analyzed by both the network and five testability experts. The test revealed a 95% agreement between the network and the experts.

In addition to the neural network, the activation values of the network were kept, and trending analysis was performed. If we consider $\Omega(t)$ to be the activation value of the network at time t, then we can consider a sequence of activation values, $\Omega(t_i)$, ..., $\Omega(t_j)$. If the values in this sequence are such that

$$\Omega(t_i) \leq \Omega(t_{i+1}) \leq \ldots \leq \Omega(t_{j-1}) \leq \Omega(t_j),$$

then it is clear no progress is being made toward finding an answer. As a result, POINTER terminates fault isolation. As a final termination criterion, given that the first two steps fail to terminate the process, POINTER terminates fault isolation in the event every test in the model has been executed n times or has been labeled untestable or unavailable. In this case, n may vary, depending on the base confidences associated with each respective test.

EXAMPLE

As an example of how one might use this approach to reasoning under uncertainty, we use the simple serial system given in Figure 1. For this example, we perform tests t_1 and t_3 and assume that t_1 passes but t_3 fails. If this is all of the testing we perform, we would find that components c_2 and c_3 are ambiguous (i.e., at least one of these two components is bad). Now suppose that t_1 has a 0.8 confidence associated with it that it will pass. This results in the support and plausibility values listed in the top third of Table 1. In addition, if we assume that the confidence in t_3 failing is 0.9, we have the support and plausibility values given in the second third of Table 1.

To illustrate the effect of conflict, suppose we retest t_3 and the test passes with a confidence of 0.8. In this case, if we assume that the current hypothesis is c_2 and c_3 (assume t_2 is untestable), then we find that the test denies all members of the hypothesis set **H**. Thus, we have a conflict. This results in the support values rising for the unanticipated

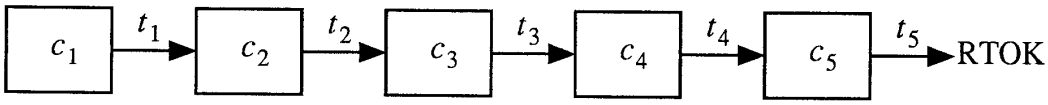

Figure 1. A Simple System for Example Purposes

Table 1. Evidential Values for Example

Component	Support	Plausibility	Probability*
	t_1 = Pass (confident 0.8)		
c_1	0.00	0.20	0.037
c_2/c_3	0.16	1.00	0.241
c_4	0.16	1.00	0.241
c_5	0.16	1.00	0.241
RTOK	0.16	1.00	0.241
UR	0.00	1.00	0.000
	t_3 = Fail (confidence 0.9)		
c_1	0.13	0.60	0.211
c_2/c_3	0.37	1.00	0.362
c_4	0.03	0.55	0.142
c_5	0.03	0.55	0.142
RTOK	0.03	0.55	0.142
UR	0.00	1.00	0.000
	t_3 = Pass (confidence 0.8)		
c_1	0.06	0.47	0.127
c_2/c_3	0.24	0.73	0.256
c_4	0.08	0.70	0.179
c_5	0.08	0.70	0.179
RTOK	0.08	0.70	0.179
UR	0.20	1.00	0.080

*The probability values are normalized to sum to 1.0.

result and each of the components downstream of t_3, and the support values decreasing for each of the components upstream of t_3, as given in the bottom third of Table 1.

SUMMARY

An approach to reasoning under uncertainty exists that incorporates methods from fuzzy logic, Dempster-Shafer evidential reasoning, and neural networks. We were concerned with developing an approach that behaves well and readily adapts to the needs of the user while maintaining a friendly user interface. The fuzzy logic qualifiers permit a technician to provide confidence information without understanding confidence values or probability. The modified Dempster-Shafer approach permits multiple measures of certainty to be maintained and includes a straightforward conversion to a Bayesian interpretation. Finally, the neural network in combination with a simple trend analysis identifies when an answer has been obtained, thus signaling POINTER to display the results to the technician with confidence indicators. We also have reversed the process on the back end, converting confidence values back into fuzzy qualifiers, again avoiding the necessity for the user to become familiar with probability terminology. The result is a fully integrated, multidisciplinary approach to reasoning with incomplete and uncertain data.

REFERENCES

[1] Simpson, W. R., "STAMP Testability and Fault Isolation Applications 1981-1984," *AUTOTESTCON 1985 Symposium Proceedings*, Uniondale, Long Island, New York, October 1985.

[2] Esker, E. A., "Testability Analysis: Applications in the Real World," *Proceedings of the IEEE Integrated Diagnostics Symposium*, Dayton, Ohio, December 1985.

[3] Simpson, W. R., "Testability and Fault Diagnosis of Airline Avionics," special edition of *PLANE TALK*, AMC Open Forum, Houston, Texas, March 1983.

[4] Simpson, W. R., and B. A. Kelley, "Multidimensional Context Representation of Knowledge-Base Information," *Proceedings of 1987 Data Fusion Symposium*, Laurel, Maryland, June 1987.

[5] Kelley, B. A., and W. R. Simpson, "The Use of Information Theory in Propositional Calculus," *Proceedings of 1987 Data Fusion Symposium*, Laurel, Maryland, June 1987.

[6] Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, New Jersey, 1976.

[7] Dempster, A. P., "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society*, Series B, 1968, pp. 205-247.

[8] Simpson, W. R., and J. W. Sheppard, "The Application of Evidential Reasoning in a Portable Maintenance Aid," *AUTOTESTCON 1990 Conference Record*, San Antonio, Texas, September 1990.

[9] Simpson, W. R., and J. L. Graham, "Notes on Evidential Reasoning," STAMP Technical Note No. 0343, ARINC Research Corporation, August 1989.

[10] Sheppard, John W., and William R. Simpson, "A Neural Network for Evaluating Diagnostic Evidence," *Proceedings of the National Aerospace Electronics Conference*, Dayton, Ohio, May 1991.

[11] Zadeh, L. A., "Possibility Theory and Soft Data Analysis," in L. Cobb and R. M. Thrall (eds.), *Mathematical Frontiers of the Social and Policy Sciences*, Westview Press, Boulder, Colorado, 1981, pp. 69-129.

[12] Kandel, A., *Fuzzy Techniques in Pattern Recognition*, John Wiley and Sons, New York, 1982.

[13] Rumelhart, D. E., G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge, Massachusetts, 1986, pp. 318-362.

[14] Shannon, C. E., "A Mathematical Theory of Communications," *Bell Systems Technical Journal*, Vol. 27, 1948, pp. 379-423.