Maintaining Diagnostic Truth with Information Flow Models

John W. Sheppard
ARINC
2551 Riva Road
Annapolis, MD 21401
sheppard@arinc.com

Abstract–**Most diagnostic systems today, whether applying a static test strategy or generating the strategy incrementally, assume a fixed configuration for the system being tested. Complex systems, including communications networks and adaptive control systems, may undergo many configuration changes during normal operation and test. In this paper, we present an approach to modeling system dynamics and coupling the dynamics to a diagnostic model for on-line test and diagnosis. We focus on the problem of maintaining current truth values as the diagnostic context changes and provide efficient algorithms that can be readily incorporated into information flow model-based diagnostic systems.**

I. INTRODUCTION

In most maintenance and diagnosis problems, one can assume that the test subject may be tested in a fixed configuration under tight control. In such an environment, it is possible to construct static models of the test subject (e.g., behavioral models, structural models, or heuristic rules) that are manipulated and processed with the results of several tests. Unfortunately, in highly dynamic systems in which the visibility and accessibility of test information varies with the operation of the system, these static models prove insufficient to diagnose problems as they arise.

The *information flow model* [1], the model developed in the System Testability and Maintenance Program (STAMP®) and processed by the Portable Interactive Troubleshooter (POINTER™) is a static model of a test subject that allows drawing conclusions about the test subject from test information (a process called *inference*). This model primarily consists of the inference relationships between sets of tests and sets of possible diagnoses that can be made about the test subject. Unfortunately, POINTER and the information flow model are typical of knowledge-based diagnostics today in their inability to address the dynamics of the test subject and the impact of these dynamics on diagnostic information.

The goal of this paper is to identify an approach to process the information flow model with new diagnostic algorithms capable of handling dynamically changing environments. Previous research in fault isolation in a dynamic environment has focused on the importance of understanding system configuration, but no clear approaches to adapting the diagnostic process with a configuration change have been identified. In particular, we have not been able to identify the results of any research that specifically addressed the question, "What happens if the system configuration changes in the middle of a diagnosis?" Some research; however, has suggested some possible approaches.

Bouloutas, Hart, and Schwartz [2] and Wang and Schwartz [3] assume that the system being tested can be represented as a set of finite state machines (FSMs), and they focus on the issue of fault detection. The approach proposed in [2] involves constructing an "observer" FSM such that the observer will be able to detect when a fault occurs. Unfortunately, constructing these observer FSMs is not straightforward and unlikely to be practical in a real-time environment.

In other research, several investigators appear to be applying fault isolation techniques that are very similar to the process used in STAMP and POINTER. For example, Mannione and Paschetta [4] describe an expert system for the Italian Telecommunications Network whose rules are constructed by combining information from a fault dictionary-like data structure and a topological database. As the topology changes, the faults rule base (i.e., the fault dictionary) is modified to reflect the new connectivity. This faults rule base appears to be concerned with tracking the propagation of alarm information through the network.

II. THE INFORMATION FLOW MODEL

To address the problems associated with performing system diagnosis and analyzing system testability, we introduced the concept of an information flow model [1]. This model-based approach to system test and diagnosis incorporates techniques from information fusion and artificial intelligence to guide analysis. The model represents the problem to be solved as information flow. Tests provide information, and diagnostic inference combines information from multiple tests using information fusion and statistical inference.

The structure of the information flow model facilitates our ability to compute testability measures and derive diagnostic strategies. An information flow model has two primitive elements: *tests* and *fault-isolation conclusions*. Tests include

0-7803-3379-9/96 $5.00

any source of information that can be used to determine the health state of a system. Fault isolation conclusions include failures of functionality, specific non-hardware failures (such as bus timing), specific multiple failures, and the absence of a failure indication. The information obtained may be a consequence of the system operation or a response to a test stimulus. Thus, we include observable symptoms of failure processes in the information flow model as tests. Including these symptoms allows us to analyze situations that involve information sources other than formally defined tests. Of course, the purpose of the model is to combine these information sources (tests) to derive conclusions about the system being diagnosed.

When developing a fault isolation strategy, the type, amount, and quality of test information should be considered. For our purposes, we initially assume equal quality among test results in the sense that the *good* or *bad* indication of a test actually reflects the state of the unit under test. During actual diagnosis, we relax this assumption to allow a confidence value to be associated with a test result. If all test inferences in a system are known, the information content of each test can be calculated. If a test is performed, the set of inferences allows us to draw conclusions about a subset of components. At any point in a sequence of tests, the model can be used to compute the set of remaining failure candidates. We developed a precise algorithm to look at the information content of the tests. This algorithm selects tests such that the number of tests required to isolate a fault is minimized over the set of potential failure candidates.

III. Maintaining Diagnostic Truth Under Changing Diagnostic Contexts

When the context under which diagnosis is occurring changes, any diagnostic system must maintain knowledge of what it has learned through testing consistent with the effects of these changes. At any given point, the "state" of the diagnostic system (i.e., the state of the reasoner) at a minimum consists of the vectors $\vec{T} = (val(t_1),\ldots,val(t_n))$ and $\vec{C} = (val(c_1),\ldots,val(c_m))$, where $val(t_i)$ is the known value of test t_i, $val(c_j)$ is the known value of conclusion c_j, n = the number of tests and m = the number of conclusions. At the simplest level, the diagnostic system must not reset its internal state with every change in context.

For example, suppose we have a system with five tests (t_1, t_2, t_3, t_4, t_5) and five diagnostic conclusions (c_1, c_2, c_3, c_4, c_5). Suppose in one context, only tests t_1, t_3, and t_4 are available, and they only consider conclusions c_4 and c_5. Further, suppose we have known values for tests t_1 and t_3. Next, suppose the context changes such that t_3 is no longer available and we are only able to consider conclusions c_1 and c_2. The knowledge we have gained for t_3 is still valid (provided some other event did not invalidate the knowledge) and must be maintained, even though the availability of t_3 has changed.

The issue of maintaining diagnostic truth becomes more significant when a change invalidates currently held beliefs. For example, suppose an alarm is issued in which one of several indicated faults could result in serious loss of service. For example, the alarm might indict (among other faults) a power problem on board an orbiting satellite. In this case, it may become important for the fault manager to assume that the power fault exists and attempt to "mitigate" this fault prior to fault isolating the system. Such mitigation may result in changing the configuration of the system (thus raising the dynamics issues) and may result in previous test outcomes (or alarms) being invalidated. Should this occur, the diagnostic system must modify its state in reasoning such that previously held beliefs are no longer held (and may need to be reinferred). It is possible for problems to "go away" or spontaneously clear themselves which also results in the need to revise beliefs [5]. We address such non-monotonic behavior of the system in the sections that follow.

Two approaches may be used to revise beliefs in a reasoner based on the information flow model. The first approach involves maintaining a history of the state changes through a test sequence and "reverting" to a previous step should a belief change. This approach would guarantee that the inferences made are consistent, but the time required to "reinfer" with information made available subsequent to the modified belief could be excessive. The second approach relies upon characteristics of the certainty factor approach to reasoning under uncertainty in which lost beliefs can be "uninferred." Since the certainty factor calculations are invertible, we can remove invalid inferences without the need to reinfer all subsequent information. Regardless of the approach used, it may be prudent to "flush" the reasoner's state memory periodically to remove artifacts of processing intermittent problems and false alarms [5]. These two approaches are discussed in more detail in the following sections.

A. Sequential Modification of Previous Beliefs

When we perform a sequence of tests for fault isolation, we can track the state transitions through the fault isolation session. For example, suppose we have five tests and five conclusions related as follows:

$t_1 : c_1$
$t_2 : c_1, c_2$
$t_3 : c_1, c_2, c_3$
$t_4 : c_1, c_2, c_3, c_4$
$t_5 : c_1, c_2, c_3, c_4, c_5$.

Further, assume we have the following sequence of tests and outcomes:

$t_3 = pass, t_5 = fail.$

At this point, we know the fault is either c_4 or c_5, and we still need to perform test t_4. But suppose that before we have the opportunity to perform t_4 (perhaps because of an action taken as a result of t_5 failing), we must invalidate the previous result of $t_3 = pass$. Then we must "uninfer" the results of $t_3 = pass$ which results in the set of candidate failures being c_1, c_2, c_3, c_4, and c_5 (i.e., everything but *No Fault*). Given this event, we would reset the state of belief for the diagnostic system to the previous state (i.e., the state prior to evaluating t_3). Then we would need to reapply the outcome of $t_5 = fail$ to the inference engine. We might also want to return t_3 to the inventory of available tests to perform.

We can envision this process performing as follows. With each step in the sequence of events, we store relevant state information (i.e., the set of beliefs associated with that point in the sequence). When a fact needs to be removed from the inference process, we revert to the state just prior to that fact, remove the fact from the list, and proceed through the remaining sequence. As we pass through the sequence, we reapply the outcomes and infer a new state at each step. As before, the new state is stored with the appropriate step in the sequence.

B. Incremental Modification of Previous Beliefs

The problem with tracking state transitions and "reinferring" outcomes following a change is that it may increase the computational complexity of the inference engine by as much as an order of magnitude (e.g., if the complexity of the inference engine is $O(n^2)$, this procedure could increase the complexity to $O(n^3)$). For this reason, we would prefer an approach to remove a previous inference without seriously impacting the performance of inference, and we would like to include this in the POINTER inference engine.

Currently, POINTER uses a modification of Dempster-Shafer statistical inference in its inference engine [6]. To summarize this approach, POINTER computes values for two extremes of a credibility interval for every conclusion in the model. These extremes are called *Support*, s_{c_i}, and *Plausibility*, p_{c_i}, and for a given conclusion, c_i, $s_{c_i} \le \Pr(c_i) \le p_{c_i}$. To compute these measures, we begin by assigning a confidence value to a particular test outcome, cf_{t_j}. In our formulation, we uniformly distribute the support over all conclusions supported and apply the full weight of denial (the complement of plausibility) to all conclusions denied. Thus,

$$s_{c_i} = \frac{cf_{t_j}}{|\mathbf{C}_s|} \tag{1}$$

$$d_{c_i} = cf_{t_j} \tag{2}$$

where $\mathbf{C}_s$ is the set of conclusions supported by the evidence given in t_j, and d_{c_i} is the *denial* of conclusion c_i.

From these, we compute support and plausibility measures incrementally (which is not normally the case in systems applying Dempster-Shafer) using the following sequence of steps:

$$k(t) = \sum_i \sum_j \delta_{ij} s_{c_i}(t) \tilde{s}_{c_j}(t-1) \tag{3}$$

$$\hat{s}_{c_i}(t) = \frac{s_{c_i}(t)(\tilde{s}_{c_i}(t-1) + u(t)) + \tilde{s}_{c_i}(t-1)(1 - cf_{t_j}(t))}{1 - k(t)} \tag{4}$$

$$u(t) = u(t-1)\frac{1 - cf_{t_j}(t)}{1 - k(t)} \tag{5}$$

$$\tilde{d}_{c_i}(t) = \tilde{d}_{c_i}(t-1) + d_{c_i}(t) \tag{6}$$

$$p_{c_i}(t) = 1 - \frac{\tilde{d}_{c_i}(t)}{t} \tag{7}$$

where,

$$\delta_{ij} = \begin{cases} 0; i = j \\ 1; i \neq j \end{cases}. \tag{8}$$

Note that all of these equations can be inverted. In particular, we note first that since plausibility is computed based on the current state of denial for each of the conclusions, we only need to be concerned with updating denial to derive the proper plausibility. The inverse of each of equations 5 through 7 is then,

$$\tilde{d}_{c_i}(t-1) = \tilde{d}_{c_i}(t) - d_{c_i}(t) \tag{9}$$

$$u(t-1) = u(t)\frac{1 - k(t)}{1 - cf_{t_j}} \tag{10}$$

$$\tilde{s}_{c_i}(t-1) = \frac{\hat{s}_{c_i}(t)(1 - k(t)) - u(t-1)s_{c_i}(t)}{s_{c_i}(t) + 1 - cf_{t_j}}. \tag{11}$$

Unlike the computation of plausibility, we need not worry about rederiving the normalizing constant k since it is computed based on the current support and the current estimate of total support.

Note, however, that equations 9 through 11 only reverse the computations of the Dempster-Shafer measures to the previous step in the process. Using these equations, we would need to back up to the affected state in the inference sequence and then recompute forward using the current set of valid test

results. This is unacceptable for the same reason that the sequential "uninference" process described above was unacceptable–the computational complexity of the process is increased by an order of magnitude. What we still need is the ability to revise the current set of beliefs without any significant impact on complexity. Unfortunately, for an arbitrary time step in the sequence, it is no longer true to say k, and thereby u and $\hat{s}$ need not be rederived. This is because each k and $\hat{s}$ depends upon previous values of k and $\hat{s}$. However, this is not a problem in computing denial.

Let t be the current time step and let τ be a time step ($\tau \leq t$) corresponding to an invalid fact. Then the update equation for denial becomes,

$$\tilde{d}_{c_i}(t+1) = \tilde{d}_{c_i}(t) - d_{c_i}(\tau) \tag{12}$$

So far, we have limited the discussion to correcting beliefs for the principal conclusions in the model. We have not considered how the computations need to be modified for determining the unanticipated result. Note, however, that since the plausibility of an unanticipated result is always 1.0, we only need to be concerned with updating the support measures.

The support for an unanticipated result is computed whenever evidence denies the current hypothesis. For this to occur, the evidence must deny *all* of the conclusions in the hypothesis set $\mathbf{H} \in \mathbf{C}^+$ (a non-empty set of conclusions). The amount of conflict is apportioned over the number of tests executed so far, so

$$\tilde{s}_u(t) = \tilde{s}_u(t-1) + \frac{\chi(t)k(t)cf_{t_j}(t)}{t} \tag{13}$$

where χ is the number of times a conflict has occurred. When no conflict exists, support for the unanticipated result decays according to

$$\tilde{s}_u(t) = \tilde{s}_u(t-1)\frac{t-1}{t}. \tag{14}$$

Expiring information will not affect support for the unanticipated result unless that expiration is associated with a conflicting event. In other words, if the expiring event caused no conflict originally, then $\tilde{s}_u$ just decays according to the normal decay schedule. However, if the expiring event did result in a conflict, we need to "invert" the effects of the conflict and adjust for the amount of decay that has occurred. Thus the adjusted support value (backing up one step) will become,

$$\tilde{s}_u(t-1) = \tilde{s}_u(t) - \left(\frac{\chi(\tau)k(\tau)cf_{t_j}(\tau)}{t}\right). \tag{15}$$

Also, the conflict count must be updated, thus yielding $\chi(t+1) = \chi(t) - 1$. Note, however, that, the dependence of $\tilde{s}_u$ on k also makes it impossible to arbitrarily remove a conflict.

Now consider the computation of the final support measure. First note that equation 7 computes plausibility as a function of normalized denial. Since we are interested in maintaining information on raw denial and rederiving plausibility, we have nothing to be concerned about with the plausibility calculation. However, so far there has been no equivalent normalization operation for support. At each step, support is normalized as follows.

$$\tilde{s}_{c_i}(t) = \frac{\hat{s}_{c_i}(t)(1-\tilde{s}_u(t))}{u(t) + \sum_{\forall c \in \mathbf{C}} \hat{s}_c(t)}. \tag{16}$$

So canceling the previous inference must involve subtracting out the normalized support. So,

$$\hat{s}_{c_i}(t) = \tilde{s}_{c_i}(t)\left(\frac{u(t) + \sum_{\forall c \in \mathbf{C}} \hat{s}_c(t)}{1-\tilde{s}_u(t)}\right) \tag{17}$$

Then the supports are backed out using equation 11. They do not have to be renormalized since we have rederived $\tilde{s}$ from the inversion process. Of course, the normalizer in parentheses can be stored with each test result rather than recomputing when needed.

Other issues arise when closely examining the Dempster-Shafer computations. First, it is clear that we cannot fully invert the calculations, so it appears we must implement the sequential reinference strategy at this point. We could use the one-step inversion algorithm to return to the required state, but this makes sense only when the expired test is relatively recent. Instead, we can store the state data and "jump" to the required state, skip the expired event, and reinfer forward.

Second, because of this strong dependence of the support value on previously normalized data, the Dempster-Shafer calculations exhibit a temporal-recency effect. In other words, more recent events have a greater impact on the evidential calculation than more distant events. This is significant because the evidential statistics are not temporally independent. In other words, if the same set of tests are analyzed with the same outcomes and the same confidences but in different orders, the resulting Dempster-Shafer statistics will be different.

C. An Alternative to Dempster-Shafer Inference

Because of several undesirable properties associated with incremental Dempster-Shafer inference (and traditional Dempster-Shafer inference as well), we began to explore alternative approaches to reasoning under uncertainty in which we could base our inferences on the information flow model, assign confidences to test outcomes, and perform consistent inference independent of any temporal ordering. This last property is significant in that it also permits a fully invertible inference process whereby we can "cancel" previous inferences without having to reinfer from the canceled test forward.

To guide this part of our research, we listed several characteristics that we consider reasonable for any uncertainty-based inference system. These characteristics included the following:

- We should be able to track levels of support and denial for each conclusion in the model.
- We should be able to convert these support and denial measures to an estimate of probability given the evidence, i.e., $\Pr(c_i|e)$, that is both reasonable and intuitive.
- We should be able to apply test results in any order and yield the same result.
- "Expired" test results should be removable from the inferred statistics with the same computational requirements as regular inference.
- We should be able to evaluate levels of conflict in the inference process, and all measures associated with conflict should have the same properties of any other conclusion in the model.
- Inference should be fast.

From these "requirements," we began to derive a simplified approach to reasoning with uncertain test data and discovered that we had rederived a relatively old method called *certainty factors*. Certainty factors were first used by Edward Shortliffe in his *MYCIN* system developed in the early 1970s, which provided an intuitive approach to reasoning under uncertainty in rule-based systems that had several roots in probability theory [7]. As we started to work with certainty factors, we found they satisfied all of our requirements except for the handling of conflict. The discussion that follows describes our implementation of certainty factors for the information flow model, including the creation of a conflict-management strategy that satisfies the requirements above.

As with Dempster-Shafer, we begin by noting that test outcomes either support or deny conclusions in our conclusion space. The first deviation from Dempster-Shafer is that we assign the full confidence value to all conclusions either supported or denied rather than apportioning confidence to the supported conclusions. Using the notation developed above for Dempster-Shafer, we have,

$$s_{c_i} = cf_{t_j} \tag{18}$$

$$d_{c_i} = cf_{t_j} \tag{19}$$

As before, support is only applied to a conclusion if the test outcome actually supports that conclusion, and denial is only applied if the test outcome actually denies the conclusion.

Updating support and denial over time is straightforward and has similarities to combining probabilities. In particular, we can update support and denial as follows:

$$\tilde{s}_{c_i}(t) = \tilde{s}_{c_i}(t-1) + s_{c_i}(t) - \tilde{s}_{c_i}(t-1)s_{c_i}(t) \tag{20}$$

$$\tilde{d}_{c_i}(t) = \tilde{d}_{c_i}(t-1) + d_{c_i}(t) - \tilde{d}_{c_i}(t-1)d_{c_i}(t) \tag{21}$$

According to Shortliffe, the certainty in a conclusion is given by

$$cert_{c_i}(t) = \tilde{s}_{c_i}(t) - \tilde{d}_{c_i}(t). \tag{22}$$

This is not quite enough for us since $cert_{c_i} \in [-1,1]$. First we need to rescale the value such that $cert'_{c_i} \in [0,1]$. We accomplish this as follows:

$$cert'_{c_i} = \tfrac{1}{2}(cert_{c_i} + 1). \tag{23}$$

Then we compute the probability as

$$\Pr(c_j|e) = \frac{cert'_{c_i}}{\sum_{\forall c \in \mathbb{C} \cup (unt)} cert_c}. \tag{24}$$

Note this equation includes *unt*, i.e., the unanticipated result. Later we describe how to determine certainty in the unanticipated result. For now, we emphasize that the unanticipated result is another conclusion in our conclusion space and participates in the normalization process.

Two of our requirements for the inference procedure stated that the procedure should be applicable independent of temporal order and that the procedure be invertible. In the following theorems, we prove that these two requirements are met by this method. All of the other requirements (except for the requirement to address conflict) are met by applying equations 18–24.

Theorem 1: The procedures for combining support and denial are commutative.

Proof: Since the procedures for combining support and denial are mathematically identical, we need only consider one. Without loss of generality, consider the procedure for combining support. Assume we have two values for support of some conclusion c, $s_c(t_p)$ and $s_c(t_q)$. Then we have

$$\begin{aligned} s_c(t_p \wedge t_q) &= s_c(t_p) + s_c(t_q) - s_c(t_p)s_c(t_q) \\ &= s_c(t_p) + s_c(t_q) - s_c(t_q)s_c(t_p) \\ &= s_c(t_q) + s_c(t_p) - s_c(t_q)s_c(t_p) \\ &= s_c(t_q \wedge t_p) \end{aligned}$$

□

Theorem 2: The procedures for combining support and denial are associative.

Proof: Without loss of generality, consider the procedure for combining support. Assume we have three support values for some conclusion c, $s_c(t_p)$, $s_c(t_q)$, and $s_c(t_r)$. Then we have

$$\begin{aligned} s_c((t_p \wedge t_q) \wedge t_r) &= s_c(t_p \wedge t_q) + s_c(t_r) - s_c(t_p \wedge t_q)s_c(t_r) \\ &= s_c(t_p) + s_c(t_q) - s_c(t_p)s_c(t_q) - s_c(t_p)s_c(t_r) - \\ &\quad s_c(t_q)s_c(t_r) + s_c(t_p)s_c(t_q)s_c(t_r) \\ &= s_c(t_p) + s_c(t_q \wedge t_r) - s_c(t_p)s_c(t_q \wedge t_r) \\ &= s_c(t_p \wedge (t_q \wedge t_r)) \end{aligned}$$

□

Combining these two theorems, we find we can apply any sequence of commutative and associative operators to a sequence of test results and yield the same result. Thus, the application of certainty factors is sequence independent.

Theorem 3: The procedures for computing support and denial are one-step invertible.

Proof: Without loss of generality, consider the procedure for computing support. Assume we have values for combined support and new support for some conclusion c, i.e., $\tilde{s}_d$ and s_c. Combining these two support values yields

$$\tilde{s}_c(t) = \tilde{s}_c(t-1) + s_c(t) - \tilde{s}_c(t-1)s_c(t).$$

Inverting this equation, which is equivalent to one-step inversion (i.e., "backing out" the previous value for support) yields

$$\tilde{s}_c(t-1) = \frac{\tilde{s}_c(t) - s_c(t)}{1 - s_c(t)}.$$

□

Theorem 4: The procedures for computing support and denial permit one-step inversion at any point in the evaluation process.

Proof: Without loss of generality, consider the procedure for computing support. Suppose we have a sequence of n test results with associated support values, $(s_c(t_1), \ldots, s_c(t_i), \ldots, s_c(t_n))$ and wish to cancel the effects of one of the test results on combined support. Without loss of generality, assume the test in question is t_i. Note we can rearrange our representation of the way the sequence was computed using Theorem 2 as follows:

$$\begin{aligned} \tilde{s}_c(t_1 \wedge \ldots \wedge t_i \wedge \ldots \wedge t_n) &= \tilde{s}_c(t_1 \wedge \ldots \wedge t_{i-1} \wedge t_{i+1} \wedge \ldots \wedge t_n \wedge t_i) \\ &= \tilde{s}_c((t_1 \wedge \ldots \wedge t_{i-1} \wedge t_{i+1} \wedge \ldots \wedge t_n) \wedge t_i) \end{aligned}$$

Since we could have computed the combined support by applying individual supports in any sequence, we can assume, for the sake of the inversion, that we applied the test to be canceled last in the sequence. Since we want the value of the combined support in the sequence without this last test, we have from Theorem 3,

$$\tilde{s}_c(t_1 \wedge \ldots \wedge t_{i-1} \wedge t_{i+1} \wedge \ldots \wedge t_n) = \frac{\tilde{s}_c(t_1 \wedge \ldots \wedge t_n) - s_c(t_i)}{1 - s_c(t_i)}$$

□

From Theorem 4, we find that we can eliminate any test inference through a simple, one-step inversion process, thus satisfying the requirement for an efficient procedure for "expiring" test results.

Finally, we need to consider the problem of evaluating conflict. As before, we would like the computation of support and denial for the unanticipated result, s_u and d_u, to be sequence independent and invertible. This implies that the method of determining conflict based on test results denying the current hypothesis is not valid since the derivation of the hypothesis is context dependent. We need an approach to determine conflict that will be tied to the tests themselves rather than to the state of inference.

This requirement leads to the following observation. Recall that all test outcomes support some conclusions and deny other conclusions. Prior to doing any diagnosis, we can determine the support sets for each of the tests. Determining the denial set is done by taking the complement of the support set which adds no new information to our calculation. Further, the impact of denial is based on a single failure assumption which makes determining conflict based on denial questionable.

For any given test, we can determine the test's support set when the test passes and when the test fails. We want to compare these support sets to the support sets of other tests. In particular, for a sequence of tests, we are interested in determining the relative conflict between all pairs of tests in that sequence. Support and denial for conflict then consist of combining support and denial at each step in the sequence using the combination procedures described by Equations 20 and 21. All we need now is a way to determine s_u and d_u.

Consider two tests t_i and t_j. These two tests may conflict in any of four possible situations–when both tests pass, when both tests fail, when t_i passes and t_j fails, and when t_i fails and t_j passes. Without loss of generality, suppose both tests fail. If we consider the intersection of the tests' support sets given they fail, we claim that if the intersection is the empty set, these two outcomes are inherently conflicting (i.e., they support completely different sets of conclusions and, in fact, deny each other's sets of conclusions). In this scenario, we can determine the relative amount of conflict as follows:

$$\chi(val(t_i) = \text{FAIL} \wedge val(t_j) = \text{FAIL}) = 1 - \frac{\left|\mathbf{C}^{\mathrm{F}}_{t_i} \cap \mathbf{C}^{\mathrm{F}}_{t_j}\right|}{|\mathbf{C}^{\mathrm{F}}_{t_i} \cup \mathbf{C}^{\mathrm{F}}_{t_j}|} \quad (25)$$

where $\mathbf{C}^{\mathrm{F}}_{t_i}$ is the set of conclusions supported by t_i failing.

Similarly, we can determine the relative amount of conflict *denial* associated with a pair of test outcomes. If the intersection of the support sets is not empty, then there exists a set of conclusions mutually supported by these two test outcomes. This area of mutual support indicates that the test outcomes are inherently non-conflicting, thus indicating we can deny the presence of conflict in the diagnostic process. Therefore, we can compute the relative denial of conflict between two test outcomes as follows:

$$\bar{\chi}(val(t_i) = \text{FAIL} \wedge val(t_j) = \text{FAIL}) = \frac{|\mathbf{C}^{\mathrm{F}}_{t_i} \cap \mathbf{C}^{\mathrm{F}}_{t_j}|}{|\mathbf{C}^{\mathrm{F}}_{t_i} \cup \mathbf{C}^{\mathrm{F}}_{t_j}|}. \quad (26)$$

Individual values for s_u or d_u depend on the confidence in the test outcomes and can be computed as

$$s_u(val(t_i) \wedge val(t_j)) = cf_{t_i} cf_{t_j} \chi(val(t_i) \wedge val(t_j)) \quad (27)$$

$$d_u(val(t_i) \wedge val(t_j)) = cf_{t_i} cf_{t_j} \bar{\chi}(val(t_i) \wedge val(t_j)) \quad (28)$$

As tests are evaluated, we accumulate support or denial for the unanticipated result in a similar fashion to equations 20 and 21 except that a single test outcome can cause several new "events" to be added. Formally, we perform the accumulation as follows:

$$\tilde{s}_u(\tau) = \left(\bigoplus_{i=1}^{\tau-1} s_u(val(t_i) \wedge val(t_\tau)) \right) \oplus \tilde{s}_u(\tau - 1) \quad (29)$$

$$\tilde{d}_u(\tau) = \left(\bigoplus_{i=1}^{\tau-1} d_u(val(t_i) \wedge val(t_\tau)) \right) \oplus \tilde{d}_u(\tau - 1) \quad (30)$$

where $\oplus$ denotes combination as defined in equations 20 and 21. Inversion works analogously to Theorem 4, except all pairwise combinations between the canceled test and other tests in the sequence must be considered.

IV. Modeling Time/Event Sensitive Truth

Now that we are able to modify beliefs when previously known facts become invalid, we need to consider the situations under which facts can be invalidated. The motivation behind this discussion arises from configuration changes invalidating previously known facts, but this is only one potential cause. In general, we can categorize causes of invalidation to be time- or event-based.

A. Time Sensitive Truth

Truth may expire after some period of time. We may wish to consider cases in which truth decays over time or all events expire after a fixed time. Alternatively, we may wish to apply expiration times to specific facts (i.e., test outcomes). First we consider the case where facts expire. Note that the global expiration is just a special case of the local expiration in which all test outcomes have the same expiration. Therefore, we will only consider the case where an expiration time has been associated with a specific test or test outcome.

An expiration time can be associated with any test or test outcome. When we define an expiration time, the diagnostic system needs to monitor an active clock and compare this clock against the current set of time-sensitive facts to see if a fact expires. The best way to accomplish this without bogging down the processor in a busy-wait loop is by only checking to see if time thresholds have been crossed when an event requiring the inference engine to run occurs. For example, we would only consider the time thresholds when we receive another test result or when the user requests a diagnosis. At that time, prior to processing the request, all of the expirations associated with time-sensitive facts would be compared against the clock. If a fact expires, the inference process would uninfer the fact using one of the processes described in section IV.

In the event we wish truth to decay (rather than expire), we need to associate a decay rate to the time-sensitive fact. Usually, such a decay rate is provided in terms of a discount factor, and the validity of the fact decays exponentially according to that rate. The best approach for handling this is to apply the discount factor to support and denial. In particular,

$$\tilde{s}_{c_i}(t + \tau) = \gamma^{\tau} \tilde{s}_{c_i}(t) \quad (31)$$

$$\tilde{d}_{c_i}(t + \tau) = \gamma^{\tau} \tilde{d}_{c_i}(t). \quad (32)$$

where $0 < \gamma \leq 1$. Of course, if $\gamma = 1$, there is no decay. Under this formalism, t is no longer based on the number of tests performed but is tied to the clock. This, then, defines a

continuous process, but we would only need to update the state of the reasoner when some event occurs as in the case where facts expire. In addition to "canceling" an inference, we need to modify the local support and denial values using the decay factors as well; otherwise, too much support or denial could be removed.

B. Event Sensitive Truth

The second situation where facts may expire arises when expiration (or loss of validity) is tied to a specific event. Events where this can occur include reconfiguration, realignment or recalibration of equipment, intermittence, or device repair. In the latter case, repair would result in state reset if there are no additional faults of concern. Otherwise, all of the test results associated with that fault would need to be invalidated.

Modeling events that can invalidate truth can be more complicated than the time-based expiration or decay of truth. Where the computation occurs in the same way as with an expiration, and detection of an event is straightforward, we need a way to represent these events and tie them to specific facts that may exist in the knowledge base. A general formalism would be to create an entity in the diagnostic model of the form,

```
event(EventID,FactList)
```

where `EventID` uniquely identifies some event that can be detected in the system and `FactList` identifies all of the facts that would be invalidated should the event occur. Then, whenever the event occurs, the list of facts associated with that event would be compared to the list of facts in the reasoner state, and should a match occur, the associated facts would be invalidated using one of the processes in section IV.

V. SUMMARY

In this paper, we present the results of investigating the problem of fault isolation in a dynamic environment. In particular, we were concerned with the problems of adapting models and inferred beliefs as system state changes. Both problems were addressed in the context of the information flow model-based approach to fault diagnosis.

Inference procedures needed to be modified to address the concern of test results being invalidated. The current implementation of the Dempster-Shafer inference engine in POINTER is sequence dependent, thus making the process uninvertible (except by completely reversing the sequence). To remedy this problem, we propose an alternative paradigm based on Shortliffe's certainty factors in which we couple a conflict management mechanism with the certainty factor formulation. Our approach offers advantages with respect to certainty factors in that the inferences are one-step invertible, associative, and commutative. These properties permit inferences to be removed arbitrarily, i.e., no sequence dependence exists in the inference procedure.

ACKNOWLEDGMENTS

The work reported in this paper was performed under ARINC independent research and development. As such, we appreciate the assistance of other members of the research team–Terry Vines, Brian Pickerall, Jeff Curie, and John Liccione. Finally, we appreciate the many helpful discussions with William R. Simpson, Don Gartner, and Tim Wilmering in the early stages of this work.

REFERENCES

[1] Sheppard, J. W. and W. R. Simpson. 1991. "A Mathematical Model for Integrated Diagnosis," *IEEE Design and Test of Computers*, Vol. 8, No. 4, pp. 25-38.

[2] Bouloutas, A., G. W. Hart, and M. Schwartz. 1992. "Simple Finite-State Fault Detectors for Communication Networks," *IEEE Transactions on Communications*, Vol. 40, No. 3, New York: IEEE Press, pp. 477-479.

[3] Wang, C. and M. Schwartz. 1993. "Fault Detection with Multiple Observers," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 1, New York: IEEE, pp. 48-55.

[4] Manione, R. and E. Paschetta. 1994. "An Inconsistencies Tolerant Approach in the Fault Diagnosis of Telecommunications Networks," *Proceedings of the IEEE Global Conference on Communications*, New York: IEEE Press, pp. 459-469.

[5] Sutter, M. T. and P. E. Zeldin. 1988. "Designing Expert Systems for Real-Time Diagnosis of Self-Correcting Networks," *IEEE Network*, New York: IEEE Press, pp. 43-51.

[6] Simpson, W. R. and J. W. Sheppard. 1994. *System Test and Diagnosis*, Boston, Massachusetts: Kluwer Academic Publishers.

[7] Shortliffe, E. H. 1976. *Computer-Based Medical Consultations: MYCIN*, Mew York: American Elsevier Publishing, Co..