MULTIPLE FAILURE DIAGNOSIS

John W. Sheppard
ARINC Research Corporation
2551 Riva Road
Annapolis, MD 21401
Phone: (410) 266-2099
E-Mail: sheppard@arinc.com
Fax: (410) 266-4010

William R. Simpson
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, Virginia 22311-1772
Phone: (703) 845-6637
E-Mail: rsimpson@ida.org
Fax: (703) 845-6788

Abstract- **Model based diagnostic systems have generally avoided the issue of multiple failure diagnosis due to the computational complexity of covering all possible multiple faults and still providing an efficient diagnostic strategy. Optimization of decision trees is already known to be *NP*-complete, and the number of combinations of multiple faults just serves to exacerbate the problem. Nevertheless, model based diagnosis is becoming popular, and the need for multiple failure diagnosis is real. In this paper, we will provide a formal analysis of the multiple failure problem in the context of one model based approach. Specifically, we will discuss algorithms and their complexity for diagnosing multiple failures using the information flow model.**

I. INTRODUCTION

Recent years have seen the proliferation of a wide variety of model-based diagnostic systems. ARINC developed the *System Testability and Maintenance Program* (*STAMP®*) in 1981 to provide a tool for modeling diagnostic information in complex systems and assessing the system testability [1]. In 1988, the *Portable Interactive Troubleshooter* (*POINTER*™) was developed to process models from STAMP for interactive diagnosis [2]. The model used by *STAMP* and *POINTER* is called the *information flow model* (described in section 2) and has been described in detail in [3]. Early forms of the information flow model are *logic model* [4], *causal model* [5], or *dependency model* [6] and are processed by such systems as *LOGMOD* (Logic Model) [4], *STAT* (System Testability Analysis Tool) [7], *WSTA* (Weapon System Testability Analyzer) [8], and *ADS* (Adaptive Diagnostic System) [9].

This paper presents a discussion on multiple failure diagnosis using the information flow model. The information flow model is a *failure-oriented* diagnostic model in which specific failure modes of a system are specified as the possible conclusions to be drawn. Tests detect faults when they are present in the system. The diagnostic task proceeds by proving subsystems in the system have not failed. Diagnosis results by examining the set of candidates remaining after all test outcomes are known.

From this discussion we can envision multiple tests detecting the same faults and the failed test outcomes *inferring* the presence of a fault. Further, we may be able to directly infer the outcomes of tests based on previous testing thus leading us to a notion of diagnostic information flowing from a fault through a set of tests. This flow corresponds to inference traces that may result from testing.

Common to most of the systems mentioned above is the assumption that diagnosis proceeds from a single failure assumption. One notable exception is *STAT* which is able to perform multiple failure diagnosis by processing a *success-oriented* diagnostic model. Such a model tends to contradict the notion of diagnostic information flow. Tests do not detect failure; tests verify functionality. Unfortunately, the algorithms used by *STAT* are proprietary, so a description of the approach used for multiple failure diagnosis is unknown.

II. THE INFORMATION FLOW MODEL

To address the problems of performing system diagnosis and analyzing system testability, we introduced the concept of an information flow model [10]. 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 symbolic logic and pattern recognition.

The structure of the information flow model facilitates our ability to formulate testability measures and derive diagnostic strategies. An information flow model has two primitive elements: *tests* and *fault-isolation conclusions*. Tests include 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. The purpose of the model, of course, is to combine these information sources (tests) to derive conclusions about the system being diagnosed.

The model also includes three special primitive elements: *testable inputs*, *untestable inputs* and *No Fault*. The inputs represent information entering the system that may have a direct bearing on the health state of the system. A testable input is a conclusion corresponding to an external stimulus combined with a test that examines the validity of that stimulus. If we have an input that cannot be examined for validity, that element is called an untestable input. Finally, the model includes a special conclusion corresponding to the condition that the test set found no fault. The No Fault conclusion, also sometimes referred to as RTOK (for retest okay), provides us with a closed-set formulation that includes anything not directly accounted for.

After specifying the primitive elements of the model, the next step is to determine the logical relationships among the tests and between the tests and the conclusions. The basic representation of the information flow model includes a logical representa-tion of the system being analyzed. In this representa-tion, we define logical values for tests and fault-isolation conclusions. Specifically, if a test fails, it has a logic value of *true*; if a test passes, it is *false*. An asserted conclusion is *true*; a conclusion elimin-ated from consideration is *false*. To determine the logical relationships, an analyst will consider the following for each test:

- What inferences can be drawn from observing a test failing?
- What inferences can be drawn from observing a test passing?

In the initial stages of modeling, the first issue is more important. The modeler is interested in listing conclusions that, corresponding to a failure, would explain the considered test failure. The modeler is also interested in listing tests that, should they fail, would imply that current test fails. If such tests do exist, we say that the current test *observes* tests and conclusions that may cause it to fail. That is, a observational relationship exists. The second question is important in determining the type of test (e.g., whether the information provided is symmetric). This type of model is also referred to as a causal model [5] and [11].

The notion of observation is limited. Because of the way we have defined observance, we are limited to a logical interpretation where if a given conclusion is to be drawn (i.e., the corresponding element in the system has failed), then all tests observing that conclusion are also true. This relationship is represented in the following logical form:

$$f_i \rightarrow \bigwedge_j t_j \qquad (1)$$

where test t_j observes conclusion or fault f_i, and $\wedge$ represents logical conjunction. This form is also true if we know that a test has failed. In this case, all tests observed by the failed test must also fail.

$$t_i \rightarrow \bigwedge_j t_j \qquad (2)$$

given t_j observes t_i.

Observation also provides information about the possible cause of a test failure. If a test passes, then all elements which the test observes (both tests and conclusions) must also pass. Thus,

$$\sim t_i \rightarrow \left(\bigwedge_j \sim t_j\right) \wedge \left(\bigwedge_k \sim f_k\right) \qquad (3)$$

given t_i observes t_j and f_k.

We may, however, want to have the correspond-ing logical expressions with the connectives reversed:

$$f_i \rightarrow \bigvee_j t_j \qquad (4)$$

$$t_i \rightarrow \bigvee_j t_j \qquad (5)$$

$$\sim t_i \rightarrow \left(\bigvee_j \sim t_j\right) \vee \left(\bigvee_k \sim f_k\right) \qquad (6)$$

where $\vee$ represents disjunction.

The information flow model formulation does not directly handle equations 4-6. Equations 4 and 5 result when a fault *might* lead to the failure of one or more other tests. We call the set of tests that may fail a test-disjunct set. Equation 6 provides for the inclusion of multiple conclusions in the model (the subject of this paper). We derived the basic formulation to limit the combinatorial growth of the search space. However, it is important to include the logical relationships because they are a part of real systems. In this paper, we will address only Equation 6 by describing how to infer multiple conclusions from the single conclusion based model. A detailed discussion of modeling these three equations is given in [3].

III. DEFINITIONS

In preparation for developing algorithms for multiple failure diagnosis with the information flow model, we begin with several formal definitions. The definitions are intended to formalize the discussion in the previous section.

Definition 1. S is a system to be diagnosed.

The definition of system has been problematic for many researchers exploring system level diagnostics. The definition we use is intentionally very broad so as to allow analysis at any required level of detail. Thus a system can consist of an aircraft, a radar unit, a power supply, a chip or set of chips, or one or more active or passive components on a card. Systems need not be limited to physical items nor to electrical items. In fact, this is made clear by the next definition.

Definition 2. $\boldsymbol{F}$ is a set of faults to be detected and isolated where $f_i \in \boldsymbol{F}$ is one fault in this set of faults. Also, $\boldsymbol{F} \subseteq \boldsymbol{S}$. The value of f_i ($val(f_i)$) $\in$ {present, absent}.

Notice that the definition of a set of faults is related to the definition of a system. Thus we say a system is a collection of faults to be isolated. This is the basis of the failure-oriented model described in section 1. Also, since we are concerned with the detection and isolation of faults we are really trying to draw conclusions about the health of the system.

Given the assumption that tests provide information about the failure characteristics of the system by either indicting or exonerating possible failure modes, we formally define the set of tests as follows.

Definition 3. $\boldsymbol{T}$ is a set of tests to be used to detect the presence of faults $f_i \in \boldsymbol{F}$, and $t_i \in \boldsymbol{T}$ is a test such that $val(t_i)$ $\in$ {pass, fail, unknown}.

Given this definition, we do not include the parameterization (i.e., the formal specification) of the test in the information flow model. This is abstracted out of the model by providing a mapping from the test measurements (i.e., outputs) to the test outcome. We need not limit the outcomes to the set indicated in the definition, but all tests can be transformed into tests (or combinations of tests) with exactly this set of outcomes. For simplicity, we will use the notion of a binary outcome test extended to include value of unknown.

The task before us can be stated simply. Given a set of tests $\boldsymbol{T}$ and a set of possible faults $\boldsymbol{F}$ in a system $\boldsymbol{S}$, determine which faults are present in the system. For any $f_i \in \boldsymbol{F}$, we want to know if $val(f_i)$ = present. Since certain tests detect the presence of a fault, we can examine the complete set of test outcomes to determine the presence or absence of a fault. In other words, the presence of a fault is completely determined by the value assignments of the set of tests. This value assignment is called a *diagnostic signature* and is defined formally as follows.

Definition 4. $\boldsymbol{D}$ is the set of diagnostic signatures where a diagnostic signature $D_i \in \boldsymbol{D}$ is the set of ordered pairs $(t_i, v_i) \in \boldsymbol{T} \times \boldsymbol{V}$ such that $|D_i| = |\boldsymbol{T}|$, $v_i \in \boldsymbol{V}$, and $v_i \in$ {pass, fail}.

Given a set of diagnostic signatures, the task is to map each signature to a fault and vice-versa. Indeed, this is easily formalized as follows.

Definition 5. A diagnosis Δ is a one-to-one mapping such that Δ: $\boldsymbol{D} \leftrightarrow \boldsymbol{F}$ (i.e. Δ: $\boldsymbol{D} \rightarrow \boldsymbol{F}$ and Δ^{-1}: $\boldsymbol{F} \rightarrow \boldsymbol{D}$).

Definition 6. A observational relationship δ exists between test t_i and fault f_j iff a) the presence of f_j results in $val(t_i)$ = 'fail', and b) $val(t_i)$ = 'pass' indicates f_j is not present.

Definition 7. A observational relationship δ exists between test t_i and test t_j iff a) $val(t_j)$ = 'fail' indicates $val(t_i)$ = 'fail', and b) $val(t_i)$ = 'pass' indicates $val(t_j)$ = 'pass'.

The set of faults $\boldsymbol{F}$, tests $\boldsymbol{T}$, and diagnostic signatures $\boldsymbol{D}$ are sufficient to define the core of the information flow model (known more commonly as a diagnostic inference model). Note that the set of tests $t_i \in \boldsymbol{T}$ such that for failure $f_j \in \boldsymbol{F}$, $val(t_i)$ = fail (which essentially characterizes the diagnostic signature of f_j) corresponds to the set of tests that *relates* to f_j.

IV. INFERENCE META RULES

The literature describes several inference meta rules that can be used to diagnose a system represented with the information flow model [12]. A meta rule is a rule that specifies how a system will process known information. Before we can apply these meta rules or variants of the meta rules, we note that the meta rules rely on concepts from propositional calculus. But we must show a correspondence between the infor-mation flow model and a model based on propositional calculus. It turns out this is easy to do.

Observation 1. Observational relation is equivalent to implication in the propositional calculus.

Proof. Let t_i observe f_j. Let t_i = 'pass' $\equiv$ 'false' and t_i = 'fail' $\equiv$ 'true'. Let the presence of $f_j \equiv$ 'true' and absence of $f_j \equiv$ 'false'. Implication is defined in the propositional calculus as $(A \rightarrow B) \equiv (\sim A \vee B)$. We therefore construct the following truth table using pass/fail notation.

t_i	f_j	$t_i\,\delta\,f_j$
Pass	Present	Does not hold (by definition)
Pass	Absent	Holds (by definition)
Fail	Present	Holds (by definition)
Fail	Absent	Holds (may fail from other source)

Translating the table using truth value assignments, we have

t_i	f_j	$t_i\,\delta\, f_j$
False	True	False
False	False	True
True	True	True
True	False	True

This is clearly equivalent to the truth table for $f_j \rightarrow t_i$. □

Since we have propositional calculus as the basis for the information flow model, we are in a position to begin to define the inference meta rules. The initial meta rules, as we will see, come directly from the rules available in propositional calculus. Most logic based systems rely on the availability of procedures for chaining through a set of rules. These rules correspond to implications in propositional logic, and the most common inference rules are *modus ponens*, *modus tollens*, and transitivity. Another observation immediately becomes apparent.

Observation 2. Since relates to is the same as implication, all of the traditional propositional inference meta rules (e.g., *modus ponens*, *modus tollens*, transitivity) hold.

Proof. By Definition 6.a we have "If f_j is present, then t_i fails." This can be reworded as "If f_j = 'true' the t_i 'true'." By 6.b we have "If t_i passes then f_j is absent" which can be reworded as "If t_i = 'false' then f_i = 'false'. Note that under implication, $(A \rightarrow B) \equiv (\sim B \rightarrow \sim A)$. Let A = "$f_j$ is 'true' " and B = "t_i is 'true' ". Then substituting,

(f_j is 'true' $\rightarrow$ t_i is 'true') ≡
($\sim t_i$ is 'true' $\rightarrow$ $\sim f_j$ is 'true') ≡
(t_i is 'false' $\rightarrow$ f_j is 'false').

Since we are now using propositional calculus, we can apply the standard rules of inference. This, of course, is sufficient for the proof, but the following illustrates.

a) *Modus Ponens*

$f_j \rightarrow t_i$	If f_j is present then t_i fails.
f_j	$\Rightarrow$ f_j is present.
$\therefore\ t_i$	Therefore, t_i fails.

b) *Modus Tollens*

$f_j \rightarrow t_i$	If f_j is present then t_i fails.
$\sim t_i$	$\Rightarrow$ t_i passes.
$\therefore\ \sim f_j$	Therefore, f_j is not present.

c) Transitivity

$f_j \rightarrow t_i$	If f_j is present then t_i fails.
$t_i \rightarrow t_k$	$\Rightarrow$ If t_i fails then t_k fails.
$\therefore\ f_j \rightarrow t_k$	Therefore, if f_j is present then t_k fails.

□

Unfortunately, these basic inferences rules are not expressive enough to address the complexities of diagnosis (especially multiple failure diagnosis). In a complex domain such as diagnosis, we search through a large space of possible solutions until we find the hypothesis that best explains the known information. In fault diagnosis (whether assuming single failure or multiple failure), we only need to consider $|T|$ tests and $|F|$ faults, but the number of ways we can permute the tests is extremely large (as shown in the following theorem).

Theorem 1. Given a set of $|T|$ tests, each test having m outcomes, then at most $f(m,|T|)$ test sequences can be constructed where

$$f(m,|T|) = m^{|T|}|T|!. \tag{7}$$

Proof. The proof is by induction on $|T|$. For the base case, suppose $|T| = 1$. Only one test exists for any sequence, but that test has m outcomes. Thus, we can construct m sequences. Note that $f(m,1) = m^1 1! = m$. For the inductive step, assume that the inductive hypothesis holds for test sets ranging from $|T| = 2$ to $|T| = n - 1$. Assume that $|T| = n$. Clearly, we can select any of the n tests to start a sequence, and that test has m outcomes. Thus, we have mn starts of sequences. After we choose the initial test, the subsequences are all of length $n - 1$ (or less). Therefore, by the inductive hypothesis, there are $f(m,n-1) = m^{n-1}(n-1)!$ subsequences. Combining these subsequences with the initial choice, we have

$$\begin{aligned} f(m,n) &= (mn)[m^{n-1}(n-1)!] \\ &= n[m^n(n-1)!] \\ &= m^n n(n-1)! \\ &= m^n n! \end{aligned}$$

□

From this we can extend the calculation to determine the number of diagnostic strategies that are possible.

Theorem 2. Given a set of $|T|$ tests, each test with m possible outcomes ($|T| > 1$, $m > 0$), at most $g(m,|T|)$ trees can be constructed, where assuming $g(m,1) = 1$,

$$g(m,|T|) = m^{|T|-2}|T|!. \tag{8}$$

Proof. The proof is by induction on $|T|$. For the base case, when $|T| = 1$, it is trivially true that there can be only one tree. This is a singularity in our analysis. Consider $|T|$ =

2. Now we can select either of the two tests as the root (start) of the fault tree. Each of the subtrees for the m outcomes is determined (that is, there is only one subtree for each member of **T**). This means there are only two possible trees for for $|T| = 2$. Note that

$$\begin{aligned} g(m,2) &= m^{2-2}2! \\ &= m^0 2! \\ &= 2 \end{aligned}$$

For the inductive step, assume that the inductive hypothesis holds for test sets of sizes ranging from $|T| = 2$ to $|T| = n - 1$. Assume $|T| = n$. Clearly, we can select any of the n tests as the root of the fault tree. There are then m subtrees off the root (one for each outcome). We know from the inductive hypothesis that each substree is one of $g(m,n-1) = m^{n-1-2}(n-1)!$ possibilities. Therefore, we have $(mn)m^{n-3}(n-1)! = m^{n-2}n! = g(m,n)$ possible fault trees. □

Since these two theorems include all possible permutations of test sequences and fault trees, the theorems apply directly to the case where multiple failure diagnosis is allowed. Obviously, we need to reduce the computational complexity of the diagnostic process. Many techniques for reducing complexity and generating efficient trees are presented in the literature. See for example [12] and [13]. The following discussion will assume all tests are binary outcome, but much of the discussion can be extended in a straightforward way to tests with m outcomes.

To further develop the inference meta rules available for diagnosis (with the intent of reducing complexity), we will impose an assumption that we are working in a closed universe. In other words, we assume the model completely and accurately reflects the set of possible diagnostic conclusions that can be drawn with the given test set T. Note, however, that our definition of F (the set of faults to be detected) does not appear to be complete. In particular, the absence of all faults in the system is missing. At first, it may not appear this special conclusion need be explicitly included in the model; however, we will see in a moment that adding a conclusion of no fault can be extremely valuable in processing the model.

Definition 8. $nf \in F$ is a special "fault" in which $D_{nf} \in D = \{(t_i, v_i) \mid v_i = \text{pass}\}$. This can be interpreted as a conclusion that no fault exists in S. This enables a closed world assumption.

Thus the no fault conclusion, nf, is defined by a diagnostic signature of all tests in the test set passing. Clearly, no test relates to nf; otherwise, the lack of faults in the system would require the subject test to fail. But then D_{nf} would have at least one test t_i such that $val(t_i)$ = fail. This is a contradiction.

We are now in a position to define two additional inference rules. These rules will be used to prove a theorem relating tests together based on the diagnostic signatures given in the model.

Lemma 1. Let $U = \{f_i \mid val(f_i) = \text{true}\}$. If there exists $t_i \in T$ such that $U \subseteq \{f_i \mid t_i \text{ observes } f_i\}$ then $val(t_i) = \text{true}$.

Proof. Suppose $U \subseteq \{f_i \mid t_i \text{ observes } f_i\}$ and $val(t_i) = \text{false}$. By *modus tollens* all $f_i \in U$ = false, but if $val(f_i)$ for all $f_i \in U$ = false, then no fault exists and nf = true. But then by Definition 7 and Observation 1, $nf \in U$. $\Rightarrow\Leftarrow$. □

This lemma says that if a test t_i observes U which is the current set of failure candidates (as well as other faults that have been eliminated from consideration), then t_i must fail. Note that for this rule to work, the presence of nf in the model is required. If nf is still under consideration, then no test can exist that observes *all* of the unknown (i.e., assumed failed) faults in the model. Some test must fail in order for nf to be eliminated. At that time, this lemma can only be applied to t_i if t_i observes *all* of the remaining candidate faults. Then t_i must detect the fault. Without nf in the model, it is possible that some test observes all faults at the outset thus leading to the test being inferred to fail, even though no testing has occurred.

Lemma 2. Given test $t_i \in T$, let $\xi = \{f_j \mid t_i \text{ observes } f_j\}$. If $|\xi| > 0$, $val(f_j) = \text{false}\ \forall f_j \in \xi$, then $val(t_i) = \text{false}$.

Proof. Note by Definition 5 that $\forall f_j \in \xi, \exists D_j \ni \Delta^{-1}(f_j) = D_j$. If t_i = true then $t_i \in D_j$. Further $t_i \in D_j$ precisely when t_i observes $\Delta(D_j) = f_j$. Thus $t_i \rightarrow \cup_\xi f_j$ (i.e., the disjunction of all $f_j \in \xi$). Thus $\sim\cup_\xi f_j \rightarrow \sim t_i \equiv \cap_\xi \sim f_j \in \xi \rightarrow \sim t_j$. By our assumption $\cap_\xi \sim f_j \in \xi$ is true so $\sim t_i$ is true. That is t_i = false. □

This lemma says that if a test t_i observes only faults known to be absent, then t_i must pass. This is because no other faults exist in the observed list that are candidates for failure that t_i can detect.

A common approach to constructing diagnostic models such as the information flow model is to examine the diagnostic signatures of the faults in the model and enumerate these signatures. This corresponds to modeling using *cause-effect analysis* which results in the construction of a *fault dictionary* [14]. The fault dictionary is simply an enumeration of the possible faults and all of the expected responses for the test set and is constructed through simulation of all of the faults in the system. Diagnosis using a fault dictionary frequently proceeds by evaluating all of the tests in the test set and matching the resulting signature with the signatures in the fault dictionary until the entry with the closest match is found. Using assumptions of perfect tests and single faults, the match should be exact.

Diagnosing complex systems with a fault dictionary (especially when multiple faults exist) can be very time consuming. We would like to capitalize on the availability of the inference rules described above to reduce the amount of testing. Further, we need a mechanism for identifying multiple faults within the fault dictionary. The following theorem provides an approach for identifying relationships between tests based solely on the fault dictionary. Discussions in later sections will cover the application of the inference rules for making the final diagnosis.

Theorem 3. Let $C_i = \{f_k \mid t_i \text{ observes } f_k\}$ and $C_j = \{f_k \mid t_j \text{ observes } f_k\}$. If $C_i \subseteq C_j$ then t_j observes t_i.

Proof. Since observation is identical to implication, we are trying to prove that $t_i \rightarrow t_j$. Suppose t_j = false. Then all $f_k \subseteq C_j$ are false. Since $C_i \subseteq C_j$, all $f_k \in C_i$ are also false. So by Lemma 2, t_i is false. Now suppose t_i is true. Then $\exists f_k \in C_i \ni f_k$ is true (by Definition 5). Since $C_i \subseteq C_j$, $\exists f_k \in C_j \ni f_k$ is true (Lemma 1). Thus t_j must be true by Observation 1□

Since we can identify test-to-test observations by examining the fault dictionary, we have a means for applying the standard inference rules given in Observation 2 to reduce the search space with every test measurement. This puts us in a position to develop a diagnostic procedure that is efficient and identifies the maximum possible single and multiple faults that may appear in the system (given the test set). Unfortunately, certain multiple failure conditions present severe difficulties whether or not we are able to consider all possible multiple faults. The following section describes two of these problems in detail.

V. MULTIPLE FAILURE COMPLICATIONS

Two diagnostic problems arise from the existence of multiple faults in a system that cannot be directly solved by either single-failure inference or multiple-failure inference. The first problem arises as a result of one fault masking the detection of another fault. Masking can cause serious maintenance problems when the masked fault is the *root cause* of the isolated fault. We define a root cause failure as follows.

Definition 9. Given $f_i, f_j \in T$, if f_i = true causes f_j = true, then f_i is the root cause of failure f_j.

When a root cause failure is masked by the secondary failure, then both single failure isolation and multiple failure isolation will identify the secondary failure. This is a direct result of the masking effect. Of course, if we do have a root cause failure problem, the inability to isolate the root cause leads to ineffective repair since repairing the secondary failure is futile. The secondary failure recurs when the system is reinitialized with the same symptoms as before. This places the maintenance technician in a repair circularity that will not return the system to operational status. The following Lemma formalizes this problem.

Lemma 3. Let $\zeta_i = \{t_k \mid t_k \text{ observe } f_i\}$. Given D_i and D_j, if $\Delta(D_i) = f_i$, $\Delta(D_j) = f_j$, and $\zeta_i \subseteq \zeta_j$, then detection of f_i will be masked by a failure of f_j.

Proof. Suppose f_j = true and f_i = true. Since f_j = true, $val(t_k)$ = true $\forall\, t_k \in \zeta_j$. Since f_i = true, $val(t_k)$ = true $\forall\, t_k \in \zeta_i$. Since $\zeta_i \subseteq \zeta_j$, all detections of f_i correspond exactly to detections of f_j. The converse is not true, so detection of f_i must be masked by detection of f_j. □

Thus the masking effect of a secondary failure over a root cause failure makes isolation of the root cause failure impossible without modifying the test set. Additional testing is required to prevent the masking effect from occurring.

The second problem in analyzing multiple failures is the potential for false failures. A *false failure* occurs when the symptoms of two or more faults (i.e., the combined diagnostic signatures of two or more faults) are identical to another single fault. The following provides a formal definition of a false failure. This definition is based on the concept of a *subsignature*.

Definition 10. Given a fault f_i, the *subsignature* of f_i, $SD_i = \{f_j \mid \zeta_j \subset \zeta_i\}$.

Definition 11. A false failure of fault f_i $FF_i = \{f_j \mid f_j \in SD_i\}$ where $\zeta_i = \bigcup_{f_j \in FF_i} \zeta_j$.

The existence of a false failure degrades the testability of the system. Repairing the indicated single fault has no effect, and the system is not restored to operational status. In other words, we have not identified the faulty components, and maintenance leaves the system in the original failed state. Multiple failure diagnosis does not eliminate the problem, but the situation is improved. In particular, we find that with multiple failure diagnosis, we will be able to identify both the single fault that would be falsely indicated under single failure diagnosis and the multiple fault. Unfortunately, we are unable to tell the difference between the two. We say the two diagnoses are ambiguous.

Definition 12. An ambiguity $A_i = \{f_j \mid \zeta_j = \zeta_i\}$ for some failure f_i.

Lemma 4. Given f_i with false failure indication FF_i, $FF_i \in A_i$.

Proof. By Definition 10, $\zeta_{FF_i} = \zeta_i$ which indicates an ambiguity. By Definition 11, $\zeta_i \in A_i$. Since $\zeta_i = \zeta_{FFi}$, $\zeta_{FFi} \in A_i$. □

VI. MULTIPLE FAILURE DIAGNOSIS

So far, our treatment of multiple failures has been on an intuitive level. In particular, we have treated multiple failures as the existence of one or more faults within system S. (Note that single failure diagnosis is just a special case of multiple failure diagnosis.) In developing an algorithm to do multiple failure diagnosis, we need to formalize the definition of a multiple failure. Therefore, we define multiple failure formally as follows.

Definition 13. A multiple failure $MF_i = \{f_j \mid f_j = \text{true}\}$ ($MF_i \neq \emptyset$) where $\forall f_j \in MF_i$, $f_j \in \bigcup_{\substack{f_k \in MF_i \\ f_j \neq f_k}} A_k$ and $\forall f_j, f_k \in MF_i$, $f_j \neq f_k, f_k \notin SD_j$, and $\forall FF_j \, \forall M \subset MF_i, M \neq FF_j$. MF_{isol} = the isolated multiple failure.

This definition can be interpreted as follows. A multiple failure is a non-empty set of failures in F. This is the intuitive definition, but it alone is difficult to process. Therefore, we add the following restrictions to the definition. First, each member of the multiple failure can belong to an ambiguity group, but we will only concern ourselves with one representative of that ambiguity group which we call the unique representative. (Note that an algorithm will be given later for identifying multiple failure ambiguities.) Second, for all pairs of single faults in the multiple failure, no single fault is masked by another single fault. Finally, no false failures are included in the multiple failure.

The rationale for this definition can be given in three parts. First, a multiple failure may include all combinations of members of an ambiguity group; therefore, it is sufficient to consider representative faults in an ambiguity group. Second, a multiple failure may include all combinations of members in a subsignature of a member of the group. Since members of a subsignature are masked by the primary signature, it is sufficient to consider the fault corresponding to the primary signature only. Third, a multiple failure does not include any subset comprising a false failure indication since the false failure indication is ambiguous with that subset. This third point is covered, of course, by both the first and the second restriction. Using the first restriction, the single failure (i.e., the false failure indication) represents the ambiguity.

Definition 14. $\zeta_{\text{fail}} = \{t_i \mid t_i \text{ observes } MF_{\text{isol}}\}$.

Definition 15. $D_{\text{fail}} = \{(t_i, v_i) \mid \text{if } t_i \text{ observes any } f_j \in MF_{\text{isol}}, v_i = \text{fail, else } v_i = \text{pass}$.

From Definition 14, ζ_{fail} is the set of tests that detect the presence of multiple failure MF_{isol}. From Definition 15, the diagnostic signature of MF_{isol} is given by D_{fail}. Given either ζ_{fail} or D_{fail}, we still need to determine the set of failures f_j comprising MF_{isol}. This is the general problem of multiple failure diagnosis. To reduce the size of MF_{isol} to a minimum (so as to reduce computational complexity), we want to identify the smallest set of failures necessary to generate D_{fail}. Unfortunately, the task of determining this minimum size multiple failure is also computationally complex.

Theorem 4. Finding a minimum size multiple failure matching ζ_{fail} is NP-complete.

Proof. (By reduction to minimum set covering). Because of the simple transformation to minimum set covering, this proof need not be detailed. This problem is clearly in NP since we need only specify some k and measure the size of matching with k as an upper bound. Recall that the minimum set covering problem is given as follows. Given a set **Z** and a set of subsets of **Z**, Φ. Find a minimum size subset $C \subseteq \Phi$ such that $\bigcup_{f_i \in C} f_i = \mathbf{Z}$. The correspondence to the multiple failure problem is as follows. Let $\mathbf{Z} = \zeta_{\text{fail}}$ which corresponds to the set of tests that all fail in the presence of multiple failure **Z**. Let $\Phi = \{\zeta_i \mid f_i \subseteq \mathrm{MF_Z}\}$. The task now is to find the set of failures f_i such that $\cup_i \zeta_i = \mathbf{Z}$ and the number of failures is minimized. Since each f_i is paired with a ζ_i, the task is to find the fewest (i.e., minimum ζ_i) to cover **Z**. □

Due to the NP-completeness of the multiple failure matching problem, we are now left with the task of finding a small multiple failure set that matches MF_{isol} that may not be optimal. Note that, even so, the number of multiple failures that need to be considered to generate a complete multiple failure diagnostic strategy can be quite large (as illustrated by the following two theorems).

Theorem 5. Given an arbitrary system S, there exist $|F|$ "multiple" faults, and the best case performance of multiple failure diagnosis is $O(|F| \lg |F|)$ for constructing a complete strategy.

Proof. Let S be a set of faults ordered such that for $i \geq 2$, $\zeta_i \subset \zeta_{i-1}$. This corresponds to a system of serial faults. Thus, using any multiple failure matching procedure under our stated assumptions, only single faults will be found (or ambiguous single faults, depending on the test set). S can be diagnosed with a balanced binary search tree which can be constructed in time $O(|F| \lg |F|)$. □

Theorem 6. Given an arbitrary system S, there exist $2^{|F|}$ "multiple" faults, and the worst case performance of multiple failure diagnosis is $O(|F|\,2^{|F|})$ for constructing a complete strategy.

Proof. Let S be a set of faults such that $\forall\, f_i, f_j,\ \zeta_i \not\subset \zeta_j$. This corresponds to a system of disjoint faults such as might be found in a completely parallel system. Transitivity and the application of Theorem 3 will fail to find any high order inferences since all of the signatures D_i, are disjoint. So the only applicable inference rules are *modus ponens* or *modus tollens* (Lemmas 1 and 2 will not apply until the leaves of tree, so they will not significantly reduce complexity). Also, since all D_i are disjoint all $D_i^* \in \bigcup_{D_i \in D} D_i$ are also disjoint. Thus the space of multiple failures is defined by D^* which contains $\sum_{i=1}^{|F|} \binom{|F|}{i} = 2^{|F|}$ fault signatures. Assuming we can still construct a balanced binary tree, the depth of the tree will be $\lg 2^{|F|} = |F|$. Thus the complexity of building this tree is $O(|F|\,2^{|F|})$ □

If there exists a method for anticipating the number of multiple failure sets to be isolated in a diagnostic strategy, then we may be able to limit diagnosis to some subset of multiple faults when the number of sets gets large. We know from Theorems 5 and 6 that the number of multiple failure sets for a system S ranges from $|F|$ to $2^{|F|}$.

We propose the following to estimate the complexity of the multiple failure fault tree. Let $\phi_i = |\{f_j \mid f_i \text{ masks } f_j\}|$ and denote *compl(S)* the complexity of system S. If S is serial (by Theorem 5), then *compl(S)* should equal $|F|$. On the other hand, if S is parallel (by Theorem 6), then *compl(S)* should equal $2^{|F|}$. We define *compl(S)* formally as follows.

$$\mathit{compl}(S) = 2^{|F|} - \sum_{f_i \in F} \mathit{mask}(f_i) \qquad (9)$$

where

$$\mathit{mask}(f_i) = 2^{\phi_i} - 1 \ . \qquad (10)$$

This formula clearly meets the restrictions on the bounds and appears to provide an over-estimate of a relatively small multiple when applied to models at the extremes.

VII. MULTIPLE FAILURE ALGORITHM

Given the complexity bounds on computing the set of multiple failure diagnoses and on generating a minimal multiple failure matching, we find it is necessary to approximate the optimal solution. A common approach to optimizing *NP*-complete problems is through the use of local search (called the *Greedy heuristic*). The algorithm, **MultFailFind**, is based on a greedy algorithm for minimal set covering and is given as follows.

```
Algorithm MultFailFind($\zeta_{fail}$)
    1.Let $cand = \{f_i \mid \zeta_i \subseteq \zeta_{fail}\}$
    2.Initialize $MF = \emptyset$, $\zeta_{MF} = \emptyset$
    3.Let $dist(cand_i, MF) = |\zeta_{fail}| - |\zeta_{cand_i} \cup \zeta_{MF}|$
    4.While $\zeta_{MF} \neq \zeta_{fail}$ do
        a. Let $f_{max} = \arg\max_i (dist(cand_i, MF))$
        b. $MF = MF \cup \{f_{max}\}$
        c. $\zeta_{MF} = \zeta_{MF} \cup \zeta_{f_{max}}$
    5.Return $MF$
```

This algorithm requires the definition of a distance function in which we attempt to determine how well the set of failing tests for a single conclusion (ζ_i) matches the set of tests for the multiple failure (ζ_{fail}). The single conclusion covering the most unaccounted for tests in ζ_{fail} (i.e., the furthest from the current hypothesis MF) is added to the multiple failure group in the solution (indicated by ζ_{MF}), and the procedure repeats with the remaining faults.

Recall we also wanted to be able to generate an ambiguity group for the multiple failure indicated by ζ_{fail}. The following algorithm provides a means to construct an ambiguity group (although it is not guaranteed to be complete). This algorithm can be used to identify false failures ambiguous with a single fault isolation as well.

```
Algorithm MultFailAmbig($\zeta_{fail}$)
    1.$A_{MF} = \emptyset$, $MF = \emptyset$
    2.$\zeta_{sub} = \zeta_{fail}$
    3.$cand = \{f_i \mid \zeta_i \subseteq \zeta_{fail}\}$
    4.While ($\zeta_{sub} = \zeta_{fail}$) do
        a.$MF$ = MultFailFind($\zeta_{sub}$)
        b.For all $f_i \in MF$ $cand = cand - \{f_i\}$
        c.$\zeta_{sub} = \bigcup_{f_i \in MF} \zeta_i$
        d.$A_{MF} = A_{MF} \cup MF$
    5.Return $A_{MF}$
```

The algorithm constructs an ambiguity group by applying **MultFailFind** to identify the faults matching the target signature ζ_{fail}. As members are found by **MultFailFind**, they are eliminated from the set under consideration. This leads to the incompleteness of the algorithm since single conclusions certainly can participate in several multiple failures that are ambiguous.

VIII. CONCLUSION

In this paper, we provide a formal analysis of multiple failure diagnosis using the information flow model. We limited the scope of the model to the traditional diagnostic inference model in that we required tests to have binary outcomes and the relationships to be fully symmetric. In actual diagnostic problems, these restrictions are too severe. As such, any diagnostic system extended to include multiple outcome tests and asymmetric inference must extend the analysis provided here to assess the impact of these extensions. We have already found that asymmetric inference has the potential of greatly increasing the size of a diagnostic strategy, so such extensions are likely to have a severe negative impact on the general multiple failure diagnosis problem.

We foresee several extensions to the current research that should improve multiple failure diagnosis capabilities. First, the heuristic defined in section 6 can be improved to provide tighter bounds on diagnostic complexity. Second, transitioning the multiple failure analysis into a dynamic tool rather than computing a complete strategy will eliminate the problems associated with the size of the complete diagnostic strategy since only a single path through the tree is of interest and is generated on line. Third, for problems in which the number of reasonable multiple failures is small, this approach provides a means of identifying these multiple failures. These failures can then be represented explicitly in the model to be used in a multiple failure testability analysis. To our knowledge, no tool exists that is capable of analyzing directly the impact of multiple failures on system testability.

REFERENCES

[1] William R. Simpson and Harold S. Balaban. 1982. "The ARINC Research System Testability and Maintenance Program (STAMP)," *Proceedings of the IEEE AUTOTESTCON*, New York: IEEE Press, pp. 88-95.

[2] William R. Simpson, John W. Sheppard, and C. Richard Unkle. 1989. "POINTER—An Intelligent Maintenance Aid," *Proceedings of the IEEE AUTOTESTCON*, New York: IEEE Press, pp. 26-31.

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

[4] Ralph DePaul. 1985. "Logic Modeling as a Tool for Testability," *Proceedings of the IEEE AUTOTESTCON*, New York: IEEE Press, pp.203-207.

[5] Yun Peng and James A. Reggia. 1990. *Abductive Inference Models for Diagnostic Problem-Solving*, New York: Springer-Verlag.

[6] William Keiner. 1990. "A Navy Approach to Integrated Diagnostics," *Proceedings of the IEEE AUTOTESTCON*, New York: IEEE Press, pp.443-450.

[7] Detex Systems, Inc. 1990. *Testability Through STAT*, internal documentation.

[8] J. Franco. 1988. "Experiences Gained Using the Navy's IDSS Weapon System Testability Analyzer," *Proceeding of the IEEE AUTOTESTCON*, New York: IEEE Press, pp. 129-132.

[9] Anthony Magliero. 1987. "ADS—The IDSS Adaptive Diagnostic System," *Proceedings of the IEEE AUTOTESTCON*, New York: IEEE Press.

[10] William R. Simpson, and John W. Sheppard, *System Test and Diagnosis*, Kluwer Academic Press, Boston, Mass., 1994.

[11] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. San Mateo, California: Morgan Kaufmann Publishers.

[12] William R. Simpson and John W. Sheppard. 1993. "Fault Isolation in an Integrated Diagnostic Environment," *IEEE Design and Test of Computers*, Vol. 10, No. 1, pp. 52-66.

[13] John W. Sheppard and William R. Simpson. 1993. "Performing Effective Fault Isolation in Integrated Diagnostics," *IEEE Design and Test of Computers*, Vol. 10, No. 2, pp. 78-90.

[14] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design*. New York: Computer Science Press.