Online Model-based Diagnosis for Multiple, Intermittent and Interaction Faults

Lukas Kuhn [1], **Johan de Kleer** [1], **and Juan Liu** [1]

[1] *PARC, Palo Alto, CA, USA*
{*lkuhn,dekleer,jjliu*}*@parc.com*

ABSTRACT

This paper extends model-based diagnosis (MBD) (Reiter, 1987; de Kleer and Williams, 1987) to systems which convert, move and process materials or objects. Examples of such systems are printers, refineries, manufacturing lines and food processing plants. Such plants present two challenges to model-based diagnosis: (1) the plant may process with very high speed while handling multiple objects in parallel such that retaining full details of behavior of all past objects is impractical, and (2) complex multi-way interactions can occur among components operating on the same object. We address the first challenge by synopsizing past behavior and the current knowledge in a data structure of linear size in the number of components in the system. The second challenge is addressed by introducing the notion of interaction fault. An interaction fault is present if a set of components operating on the same object, damage the object even though each component alone produces non noticeable damage. Introducing interaction faults is much simpler than introducing fine-grained models of component-object interactions. We demonstrate the approach on a highly redundant printer.

1 INTRODUCTION

Most existing approaches to model-based diagnosis presume all information flow in a system as signals. They are good for modeling systems that can be directly modeled as ODEs as is characterized by system dynamics (Shearer *et al.*, 1971). However, most real world systems transport and modify materials or objects. For example, a refinery converts one kind of fuel into another with different characteristics, a printer converts blank paper to paper with marks on it, and a food packaging line converts food and cardboard into packaged food. Such systems need to reason about both the attributes of system components (e.g., voltage, current, pressure) and the properties of handled objects (e.g., wrapped candy bar, unwrapped candy bar, partially assembled automobile). We drawn most of our experience from a high speed multi-path printing press illustrated in Figure 1 (Fromherz *et al.*, 2003). The highly redundant printing press consists of two towers each containing 2 printers (large rectangles). Sheets enters on the left and exit on the right. Dark black edges with small rollers represents a possible paper path. There are three main paper highways within the modular printer (horizontal). The printer incorporates 2 types of media handling components represented by small lighter edge rectangles. The motivation for this design is to continue printing even if some of the printers fail or some of the paper handling components fail or jam.

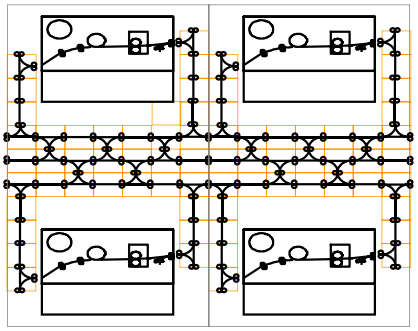

Figure 1: Model of PARC's prototype highly redundant printer.

The two main challenges of such plants are (1) continuous high speed and parallel processing which leads to a data overload with respect to retaining full details of behavior and current knowledge of all past objects, and (2) complex multi-way interactions which can occur among components operating on the same object.

Most refineries, printers, manufacturing lines run continuously and with high speed. They are expensive to halt so minor problems are ignored or compensated for by later manual processing. Unlike simple qualitative envisionments of one signal propagated through the system, we intend to address a continuous movement with large number of objects being processed at any moment. The closest analog to this type of qualitative reasoning is the parts-of-stuff ontology of (Collins and Forbus, 1987). Although more difficult to analyze, continuous high speed operation has the advantage that it is possible to gather a lot of observations.

The second distinction to classical model-based diagnosis is the fact that object or material handling systems introduce a whole new fault type: interaction faults. For example, we will often see situations where component A and component B operate without noticeable error stand-alone, yet fail when they both operate on the same object. Consider a food processing line for candy bars. There are multiple components wrapping and boxing candy bars. It may be that component A leaves a tiny rip which is of no consequence for the consumer, but boxing component B has a small protrusion such that the rip sometimes catches and destroys the candy bar. We call such faults interaction faults: A and B are perfectly operational individually but will not work correctly if A and B both operate on the same candy bar. The classical model-based diagnosis approach would be to consider both components A and B as faulted, but that is not useful for the technician. The line can be restored to full operation by either removing the protrusion on B or repairing A. There is no need to replace both A and B. Such faults occur in digital circuits as well: Gates A and B may not work well together as both may be "late". Replacing either A or B with one having an average gate delay restores the circuit to full functioning.

This paper outlines an approach for online diagnosis for multiple fault, interaction fault and intermittent fault scenarios by integrating three main components: First, we synopsize the knowledge about the past behavior in a single linear-size data structure. Necessarily some information will be lost and we rely on high throughput rates to compensate for eventual information loss. Second, we do not explicitly model the details of component-object-component behavior, but instead introduce a new generic fault category of an interaction fault. Third we adopt the meta-diagnosis abstraction framework of (de Kleer, 2007a). This idea is motivated by how technician reason about systems. A technician reasons about a system at multiple levels of abstraction. He/she will make the simplest meta-assumptions possible (e.g. single, persistent, non-interaction fault) to diagnose a system. Only when those meta-assumptions yield a contradiction will he/she choose a more detailed hypotheses model.

The resulting approach enables diagnosis for multiple-faults, interaction-faults and intermittent-faults in the context of high speed manufacturing systems.

2 RELATED WORK

Researchers in model-based diagnosis have suggested a multitude of diagnostic frameworks and notions. These are normally restricted to specific aspects of the phenomena of interest in order to facilitate modeling or computation. Example of diagnostic notions are: multiple faults (de Kleer and Williams, 1987), intermittent faults (Koren and Kohavi, 1977; de Kleer, 2007a), bridge diagnoses (Davis, 1984; de Kleer, 2007b), model-based monitoring (Dvorak and Kuipers, 1989), kernel and prime diagnoses (de Kleer *et al.*, 1992).

In addition to unifying several of the above approaches we introduce the new notion of interaction fault which are of great practical significance. We make possible the unification of all these frameworks throughout a set of algorithms and illustrate the workings of these algorithms with an example from industry (Fromherz *et al.*, 2003).

Interestingly, model-based diagnosis has been put in a common framework with the related disciplines of planning (Kuhn *et al.*, 2008), scheduling (Muscettola *et al.*, 1998), but there is a little work on combining different model-based notations in a single framework.

In the algorithmic part there has been a signifcant amount of work in model-based diagnosis. Early examples of inference algorithms are the GDE (de Kleer and Williams, 1987), the related Livingston-2 (Williams and Nayak, 1996), compilation-based approaches (Darwiche, 2001), SAT formulations (Grastien *et al.*, 2007), and many others. Our algorithms present a novel viewpoint on diagnosis as opposed to the existing single-fault or multiple-fault approaches. We split the set of components in sets, the latter labeled: good, bad or unknown and maintain a set of diagnostic foci for classifying and reclassifying components into these sets.

3 META-DIAGNOSIS

A main challenge in diagnosis is the selection of the abstraction level. If the hypothesis space is modeled to fine grain it might be possible that computation is impractical. On the other hand if we chose the abstraction level to high we might lose important details. Therefore we integrate the meta-diagnosis abstraction framework of (de Kleer, 2007a) to enable dynamic changes of abstraction level in the hypothesis space. In this approach reasoning is performed in two steps (1) first we determine the simplest meta-assumptions which don't yield a contradiction (2) followed by the second step in which we chose the corresponding algorithm (see Section 6) to determine the diagnoses with respect to the meta-assumptions. The diagnosis system contains therefore two diagnosis engines, one to determine the meta-diagnsis (which meta-assupmptions are valid) and one to determine the component diagnosis. Figure 2 illustrates the overall architecture. The decision which algorithm to determine the component diagnosis depends on meta-assumption assignment.

We formalize abstraction model-based diagnosis as: A **system** Sys as a tuple $< C, P, Z >$ where:

- C is the set of all components.
- P is a list of plans. A plan $p_i = (c_1, c_2, \ldots, c_n)$ is a sequence of components involved in the plan.
- Z is a list of observations. A observation $z_i \in \{f, s\}$ is associated with plan p_i. We denote a plan failure as f and a normal plan execution as s.

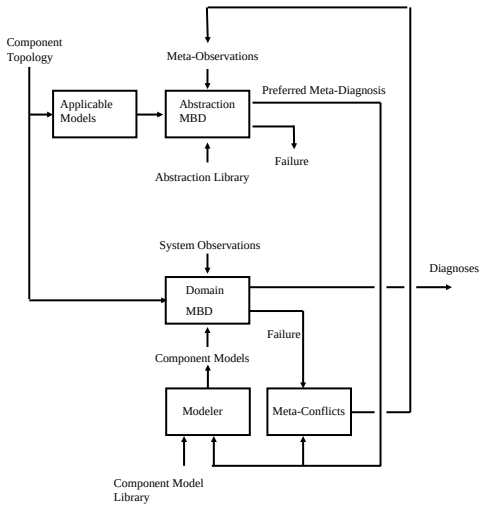

Figure 2: Architecture of an abstraction-based, model-based diagnosis engine.

An **abstraction system** A-MBD is a tuple $< SD, ABS, OBS >$ where:

- SD, constraints among possible abstractions, is a set of first-order sentences.
- ABS, the applicable abstractions, is a finite set of constants.
- OBS, a set of meta-observations, is a set of first-order sentences.

Given two sets of abstractions C_p and C_n define $D_a(C_p, C_n)$ to be the conjunction:

$$\left[\bigwedge_{c \in C_p} AB_a(c)\right] \wedge \left[\bigwedge_{c \in C_n} \neg AB_a(c)\right] \qquad (1)$$

where $AB_a(x)$ represents that the abstraction x is AB-normal (cannot be used).

A meta-diagnosis is a sentence describing one possible state of the abstraction system, where this state is an assignment of the status normal and abnormal to each abstraction.
To illustrate the idea we use three dimensions of abstraction.

- single vs. multiple faults "M".
- persistent vs. intermittent faults "I"
- stand-alone vs. interaction faults "N"

The corresponding AB_a literals are:

- $\neg AB_a(M)$: represents the abstraction that multiple faults need not be modeled.
- $\neg AB_a(I)$: represents the abstraction that the system is non-intermittent.
- $\neg AB_a(N)$: represents the abstraction that the system contains no interactions faults.

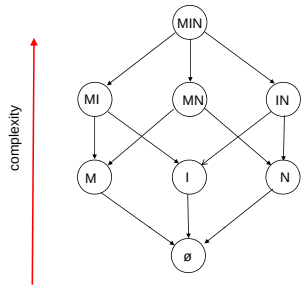

Figure 3: Meta-Diagnosis lattice: M indicates multiple faults; I indicates intermittent faults; N indicates interaction faults

4 EXAMPLE: SIMPLEST META-DIAGNOSIS FAILING

In the example we consider a system with only three components A, B, C. Initially we make three meta-assumptions : (1) the system does not have multiple faults, (2) the fault is not intermittent, (3) the fault is not interactive. This corresponds to the bottom node of Figure 3.

Suppose we observe the following plans:

time	plan	observation	conclusion
1	A,B	fail	$\neg M$ exonerates C
2	B,C	success	$\neg I$ exonerates B, C
3	A	success	$\neg I$ exonerates A

Plan 1 (A, B) fails, therefore if the system does not contain a multiple fault, one of A or B must be faulted and C cannot be faulted. Plan 2 (B, C) succeeds, therefore given the system is not intermittent B and C must be functioning correctly. Plan 3 (A) succeeds so A cannot be faulted. At this point no single fault, non-intermittent, non-interaction faults exist. This results in the meta-conflict:

$$AB_a(M) \vee AB_a(I) \vee AB_a(N). \qquad (2)$$

Analysis must consider retracting one of these three meta-assumptions. Consider multiple faults. Plan 2 exonerates B, C and plan 3 exonerates A with no dependence on the single fault assumption. Therefore, the meta-conflict is:

$$AB_a(I) \vee AB_a(N). \qquad (3)$$

Figure 4 illustrates the resulting meta-diagnosis lattice.

The system can contain either an intermittent fault or an interaction fault. For example, component A can be intermittently failing, producing a bad output at time 1 and a good output at time 3. The system can also contain an interaction fault. For example, the system can contain the interaction fault $[AB]$. We use $[...]$ to indicate the interaction fault. Plan 1 is the only plan in which A and B co-occur, therefore the interaction fault explains all symptoms.

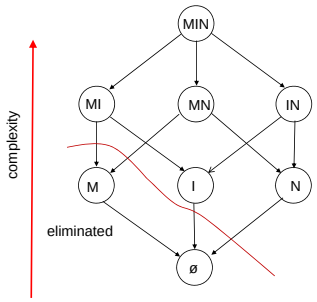

Figure 4: Meta-Diagnosis lattice after the minimal conflict $AB_a(I) \vee AB_a(N)$.

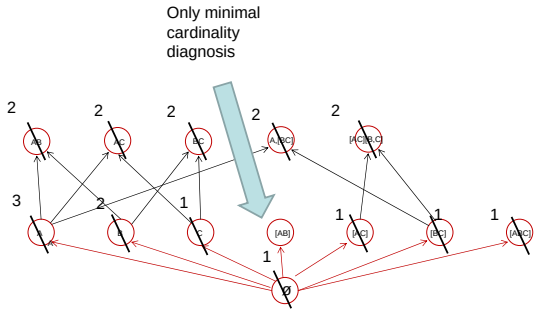

Figure 5: Fragment of diagnosis lattice for the simple 3 component system A, B, C. Includes multiple and interaction faults. The numbers indicate which plan eliminates that diagnosis.

5 META-INFERENCES

As in conventional model-based diagnosis, a tentative diagnosis is represented by the set of failing components. When a plan p succeeds the following inferences can be drawn:

- If there are no intermittent faults ($\neg AB_a(I)$), then every component mentioned in the plan is exonerated.
- If there are interaction faults ($AB_a(N)$), then every diagnosis containing a interaction fault which contains only components from p is exonerated.

When a plan p fails the following inferences can be drawn:

- Every diagnosis not containing a component in p is exonerated.

Initially, all subsets of components can be diagnoses. With the introduction of interaction faults, any combination of components can also be a fault. Therefore, if a system consists of n components, there are $O(2^{2^n})$ possible diagnoses (Eiter and Gottlob, 1995).

Figure 5 shows a fraction of the diagnosis lattice for a simple system with components three components: $\{A, B, C\}$. For simplicity we assume non-intermittent faults, but multiple and interaction faults are allowed. Consider the prior example again. Plan 1 which used A, B produced a failure. By the preceeding rules, C alone cannot explain the symptom, neither can $[AC]$, $[BC]$ or $[ABC]$. The only minimum-cardinality diagnoses are $\{A\}$, $\{B\}$ and $\{[AB]\}$. The successful plan 2 exonerates B and C. Therefore any diagnosis which contains B or C is exonerated. In addition, any diagnosis containing the interaction fault $[BC]$ is exonerated. Finally, when Plan 3 is observed to succeed, A is exonerated. The only minimum cardinality diagnosis which explains the symptoms is the interaction fault $[AB]$.

The very large size of this diagnosis lattice prompts a new diagnostic algorithm more akin to what a technician would use when diagnosing the system. It is also much more efficient for on-line diagnosis.

6 DIAGNOSTIC ALGORITHM

In this section we present a set of new diagnostic algorithms which differ from classical model-based diagnostic algorithms in significant ways. The new diagnostic algorithms maintain a set of mutually exclusive sets: diagnostic foci, good components, bad components and unknown components. Intuitively, each diagnostic focus represents a set within which we are sure there is at least one fault. Technicians will typically explore one focus at a time. As the printer or manufacturing line runs continuously there are far too many observations to record in detail. Therefore, the current foci together with the set of bad components and unknown components combined with a fixed size buffer will represent the *entire* state of knowledge of the faultedness of system components. Some information from prior observations will be discarded. The algorithms we describe may take more observations to pinpoint the true fault(s) due to the eventual information loss.

The decision which algorithm is used at any given time is determined by the meta-diagnosis abstraction framework and it's corresponding meta-assumption assignment.

6.1 Single Fault Case

The single fault case is rather simple, since we can exonerate all components after isolating the faulted component (denoted as 'DONE!' in the algorithm). However, we choose to present the single fault case to help understand the intuition and introduce our notation.

Single Fault Representation

A **system** Sys is a tuple $< C, P, Z >$ as in Section 3. A **state of knowledge** SK is a tuple $< g, b, x, su >$ where:

- $g \subseteq C$ is the set of good components.
- $b \subseteq C$ is the set of bad components.
- $x \subseteq C$ is the set of unknown components which are not under suspicion and not exonerated.
- $df \subseteq C$ is the diagnosis focus. The diagnosis focus df is a set of suspected components which cantains at least one faulted component in it.

Single Fault Algorithm

The algoritm 1 is executed for each plan and observation pair. The algorithm updates the *entire* state of knowledge of the faultedness of system components.

Consider the following example. Let Sys be a simple system with six components $C = \{A, B, C, D, E, F\}$. We assume for simplicity that we are able to execute any combination of components as a plan. Suppose component B is faulted.

In Table 1 we show for each time step t the *entire* state of knowledge.

t	p	z	g	b	x	su
0					ABCDEF	
1	ABCDE	f	F			ABCDE
2	ABC	f	DEF			ABC
3	ADE	s	ADEF			BC
4	ABDE	f	ACDEF	B		

Table 1: System with six components $C = \{A, B, C, D, E, F\}$ where B is faulted.

Algorithm 1: singleFaults(plan p_j, obs z_j)

```
if z_j == f then
    rp_j = p_j − g;
    if |rp_j| == 1 then
        b = rp_j;
        g = (g ∪ x ∪ df) − b;
        x = df = ∅;
        DONE!
    else
        if df == ∅ then
            df = rp_j;
        else
            g = g ∪ ((df ∪ rp_j) − (df ∩ rp_j));
            df = df ∩ rp_j;
            if |df| == 1 then
                b = df;
                g = (g ∪ x) − b;
                x = df = ∅;
                DONE!
else
    g = g ∪ p_j;
    x = x − g;
    df = df − g;
    if |df| == 1 then
        b = df;
        g = (g ∪ x) − b;
        x = df = ∅;
        DONE!
```

Note that every component will be a member of exactly one of the sets of the current diagnosis. Consider the sequence of plans illustrated by Table 1. Plan 1 $(ABCDE)$ fails. Therefore we focus on the fact that one of $\{A, B, C, D, E\}$ is faulted. Plan 2 (ABC) fails. Therefore, we can narrow the focus to the fact that one of $\{A, B, C\}$ is faulted, and we know that $\{D, E\}$ are good. Plan 3 (ADE) succeeds. Therefore A, D, E are exonerated. Therefore the focus narrows to $\{B, C\}$. Plan 4 $(ABDE)$ fails. Given that plan 4 intersects only in B with the suspected set, we know B is faulted (single fault assumption). All other components are exonerated.

6.2 Multiple Faults Case

In what follows we discuss an algorithm for diagnosing multiple simultaneous faults. Our algorithm allows variable amount of observation data (which can be obtained throughout execution of plans) to be retained.

Multiple Faults Representation

A **system** Sys is a tuple $< C, P, Z >$ as in Section 3. A **state of knowledge** SK is a tuple $< g, b, x, DF >$ where:

- $g \subseteq C$ is the set of good components.
- $b \subseteq C$ is the set of bad components.
- $x \subseteq C$ is the set of unknown components which are not under suspicion.
- DF is the set of diagnosis foci. A diagnosis focus $df_i \subseteq C$ is a set of suspected components with at least one faulted component in it.

Multiple Faults Algorithm with Memory

Algorithm 2 executes Procedure 3 (or for the interaction fault case Procedure 6) for each plan and observation pair. The algorithm updates the *entire* state of knowledge of the faultedness of system components. We focus on high throughput systems (100s-1000s/min) and therefore the algorithm we describe may take more observations to pinpoint the true fault(s), but it will never miss faults. We include a memory extension to mitigate the loss of diagnosis information. There are two cases in which the evaluation of an observation could lead to information loss: (1) two intersecting plans fail due to different faults or, (2) a failing plan intersects two diagnosis foci. In the first case we might not know at evaluation time if two intersecting plans fail because of the same fault or two different faults and therefore we keep the plan to later re-evaluate it. In the second case we can not extract any information before we reduce the diagnostic foci until the failing plan intersects only one diagnosis foci. Note that this might not be possible. Failing plans of either case can be helpful if they are re-evaluated later. Note that we are able to configure the memory size to address memory limitations.

Algorithm 2: Multiple (Interaction) Faults Algorithm with Memory

```
foreach p_j : P do
    if !multiFaults(p_j, z_j) then
        memorize(p_j, z_j, memorysize);
    else
        evaluateMemorizedPlans();
```

Multiple Faults Example

Let Sys be a simple system with five components $C = \{A, B, C, D, E\}$. Again we assume that we are able to execute any combination of components as a plan. Suppose component B and D are faulted.

In Table 2 we show for each time step t the *entire* state of knowledge.

Again note that every component will be a member of exactly one of the sets of the current SK. Consider the sequence of plans illustrated by Table 2. Plan 1 $(ABCDE)$ fails. Therefore we focus on the fact that one of $\{A, B, C, D, E\}$ is faulted. Plan 2 (ABC) fails. Therefore, we narrow the focus to the fact that one of $\{A, B, C\}$ is faulted, and we don't know anything about $\{D, E\}$. Plan 3 (ADE) fails. Therefore the focus narrows to A, while there may be a fault in $\{B, C\}$ (But the scope is still $\{A, B, C\}$). Plan 4 (A) succeeds. Therefore, A is exonerated. At this point we backtrack

t	p	z	g	b	x	df_1	df_2
0					ABCDE		
1	ABCDE	f				ABCDE	
2	ABC	f			DE	ABC	
3	ADE	f			DE	ABC	
4	A	s	A			BC	
5	ADE	f	A			BC	DE
6	AC	s	AC	B			DE
7	ADC	f	AC	BD	E		
8	ACE	s	ACE	BD			

Table 2: System with five components $C = \{A, B, C, D, E\}$ where B and D are faulted.

and move the focus to $\{B, C\}$. Plan 5 (ADE) fails. Therefore, given that A is exonerated, we can introduce a new focus on the fact that one of $\{D, E\}$ is faulted. Plan 6 (AC) succeeds. Therefore, C is exonerated and B is the only component left in focus 1. Therefore we know B is faulted. We close focus 1. Plan 7 (ADC) fails. Therefore, given that A, C are exonerated, D is faulted. We close focus 2 and move the remaining components (here E) in the unknown set. Plan 8 (ACE) succeeds, thus ACE is exonerated.

Multiple Faults Algorithm

Procedure 3 `multiFaults` (*plan p_j, obs z_j*)

```
if z_j == f then
    rp_j = p_j − g;
    if rp_j ∩ b = ∅ then
        if |rp_j| == 1 then
            b = b ∪ rp_j;
            foreach df_i : DF do
                if df_i ∩ rp_j ≠ ∅ then
                    x = (x ∪ df_i) − b;
                    DF.remove(df_i);
        else
            if rp_j ∩ ⋃_k df_k = ∅ then
                df_new = rp_j;
                x = x − rp_j;
            else
                foreach df_i : DF do
                    if rp_j ∩ df_i ≠ ∅ ∧ rp_j ≠ df_i then
                        if rp_j − df_i ⊆ x then
                            if |rp_j| < |df_i| then
                                x = (x ∪ df_i) − rp_j;
                                df_i = rp_j;
                            else
                                return false;
                        else
                            return false;
else
    g = g ∪ p_j;
    x = x − g;
    foreach df_i : DF do
        df_i = df_i − g;
        if |df_i| == 1 then
            b = b ∪ df_i;
return true;
```

6.3 Multiple Interaction Faults Case

Multiple Interaction Faults Representation Definition:
Let $X = \{x_1, \ldots, x_n\}$ be a set of elements.

- $P(X)$ is the power set over X, e.g. $X = \{x_1, x_2\} \leftrightarrow P(X) = \{\{\}, \{x_1\}, \{x_2\}, \{x_1, x_2\}\}$.
- $\overline{X} \equiv P(X)$ represents the power set of X.
- $\{\overline{Y}\} \sqcup \{\overline{X}\} \equiv \begin{cases} \{\overline{Y}\} & : \text{if } X \subseteq Y \\ \{\overline{X}\} & : \text{if } Y \subseteq X \\ \{\overline{Y}, \overline{X}\} & : \text{otherwise} \end{cases}$
- $\overline{P}(X) \equiv \bigcup_{Y \subseteq X} \overline{Y}$ is the set of all power sets over all possible subsets of X .
- $E(X)$ is the set of all individual components mentioned in X, e.g. $X = \{\{a, b, c\}, \{a, d, e\}, \{g\}\} \leftrightarrow E(X) = \{a, b, c, d, e, g\}$.

A **system** Sys is a tuple $< C, P, Z >$ as in Section 3.
A **state of knowledge** SK is a tuple $< g, b, x, DF >$ where:

- $g \subseteq \overline{P}(C)$ represents all global good diagnosis candidates. A diagnosis candidate is a set of components that can cause a failure. Let $X \subseteq C$ be a set of components, than $\overline{X} \in \overline{P}(C)$ represents all diagnosis candidates $dc \in P(X)$.
- $b \subseteq P(C)$ is the set of bad diagnosis candidates. $\{A, [DE]\}$ denotes that A and the diagnosis candidate $[DE]$ (interaction fault) are bad.
- $x \subseteq P(C)$ is the set of unknown diagnosis candidates which are not under suspicion.
- DF is the set of diagnosis foci. A diagnosis focus df_i is a tuple $< su_i, lg_i >$ where:
  - $su_i \subseteq P(C)$ is the set of suspected diagnosis candidates in the diagnosis focus df_i.
  - $lg_i \subseteq \overline{P}(C)$ represents all local (relevant) good diagnosis candidates.

Multiple Interaction Faults Algorithm

Procedure 4 `minimalCandidates` (*Set<Comps> C, $\overline{P}$(Set<Comps>) PC*)

```
Beginn
    CA = candidates(C, PC);
    minCar = |E(CA)|;
    MCA = ∅;
    foreach ca_i : CA do
        if |ca_i| = minCar then
            MCA = MCA ∪ ca_i;
        if |ca_i| < minCar then
            MCA = ∅;
            MCA = MCA ∪ ca_i;
    return MCA;
Ende
```

Procedure 5 `candidates` (*Set<Comps> C, $\overline{P}$(Set<Comps>) PC*)

```
Beginn
    CA = P(C);
    foreach pc_i : PC do
        CA = CA − pc_i;
    return CA;
Ende
```

Procedure 6 `multiInterFaults(`*plan* p_j, z_j`)`

```
if z_j == f then
    CA_j = candidates(p_j, g);
    if CA_j ∩ b == ∅ then
        if |CA_j| == 1 then
            b = b ∪ CA_j;
            foreach df_i : DF do
                if CA_j ∩ su_i ≠ ∅ then
                    DF.remove(df_i);
        else
            MCA_j = minimalCandidates(p_j, g);
            if CA_j ∩ ⋃_k su_k == ∅ then
                foreach c ∈ g do
                    lg_new = lg_new ⊔ (E(c) ∩ p_j)‾;
                su_new = MCA_j;
            else
                foreach df_i : DF do
                    if MCA_j ∩ su_i ≠ ∅ ∧ MCA_j ≠ su_i
                    then
                        if MCA_j ∩ ⋃_{k,k≠i} su_k == ∅
                        then
                            if |MCA_j| < |su_i| then
                                foreach c ∈ g do
                                    lg_i =
                                    lg_i ⊔ (E(c) ∩ p_j)‾;
                                su_i = MCA_j;
                            else
                                return false;
                        else
                            return false;
else
    g = g ⊔ p_j‾;
    foreach df_i : DF do
        lg_i = lg_i ⊔ (E(lg_i) ∪ E(su_i)) ∩ p_j‾;
        su_i = minimalCandidates(su_i, lg_i);
        if |su_i| == 1 then
            CA_{lg_i} = candidates(E(lg_i), lg_i);
            if |CA_j| == 1 then
                b = b ∪ CA_j;
                foreach df_i : DF do
                    if CA_j ∩ su_i ≠ ∅ then
                        DF.remove(df_i);
            else
                su_i =
                minimalCandidates(E(lg_i), lg_i);
return true;
```

As before, Algorithm 6 is executed for each plan and observation pair. The algorithm updates the *entire* state of knowledge of the faultedness of system components.

Consider the following example. Let Sys be a simple system with five components $C = \{A, B, C, D, E\}$. We assume for simplicity that we are able to execute any combination of components as a plan. Suppose component B and D are faulted. In Table 3 we show walk through the example.

Consider the sequence of plans illustrated by Table 3. Plan 1 $(ABCDE)$ fails. Therefore we focus on the fact that one of $\{A, B, C, D, E\}$ is faulted. Plan 2 (ABC) fails. Therefore, we can narrow the focus to the fact that one of $\{A, B, C\}$ is faulted, and we don't know anything about $\{D, E\}$. Plan 3 (ADE) fails. Therefore the focus narrows to A, while there may be a fault in $\{B, C\}$ (But the scope is still $\{A, B, C\}$). Plan 4 (A) succeeds. Therefore, A is exonerated. At this point we backtrack, move the focus to $\{B, C\}$ and keep A as a local (relevant) good ($\{\overline{A}\}$). The scope is now $\{B, C\}$. Plan 5 (ADE) fails. Therefore, given that A is exonerated, we can introduce a new focus on $\{D, E\}$, but we keep A as a local (relevant) good ($\{\overline{A}\}$). Plan 6 (AC) succeeds. Therefore, A, C, AC are exonerated, denoted as $\overline{AC}$. The new global goods are $\{\overline{AC}\}$, because $\{\overline{A}\} \sqcup \{\overline{AC}\} = \{\overline{AC}\}$. We update the local (relevant) goods in focus 1 to AC, because A, C, AC are relevant to focus 1. B is the only diagnosis candidate left in focus 1. Plan 7 (ADC)fails. Therefore, given that A, C are exonerated, D is faulted, because it is a minimal diagnosis candidate. We close focus 2. Plan 8 (ACE)succeeds, thus A, C, E, AC, AE, CE, ACE are exonerated, denoted as $\overline{ACE}$. The new global goods are $\{\overline{ACE}\}$, because $\{\overline{AC}\} \sqcup \{\overline{ACE}\} = \{\overline{ACE}\}$. Plan 9 (B) succeeds, thus B is exonerated, denoted as $\overline{B}$. The new global goods are $\{\overline{ACE}, \overline{B}\}$, because $\{\overline{ACE}\} \sqcup \{\overline{B}\} = \{\overline{ACE}, \overline{B}\}$. At this point we know that the diagnosis candidates A, B, C, AC relevant to focus 1 are goods. Therefore generate all minimal diagnosis candidates form the local goods $\{[AB], [BC]\}$ and move the focus to them. Plan 10 (AB)fails. Therefore, given that A, B are exonerated, $[AB]$ is faulted, because it is a minimal diagnosis candidate.

6.4 Intermittent Fault Algorithm

Intermittent faults are very common in object handling systems such as printers and manufacturing lines. It is very difficult to isolate intermittent faults which occur with low frequency but yet at high enough frequency to be unacceptable. The intermittent nature of faults brings inherent uncertainties. In this subsection, we present a statistical approach to identify highly suspected defective components. The overall integrated diagnosis framework switches to this algorithm in case the abstraction framework determines the meta-assumption $AB_a(I)$ to be true.

In the literature, diagnosis of multiple fault is often formulated as a statistical inference problem, where the hypothesis space is $X = \{00000, 00001, 00010, \cdots, 11111\}$, and each hypothesis is a bit vector indicating which components have fault. For instance $x = 00010$ means that all but the fourth component are good. Given an observation o (or an observation sequence), one can compute the posterior $p(x|o)$ via the Bayes rule:

$$p(x|o) = \frac{p(o|x)p(x)}{p(o)},$$

where $p(o|x)$ is the observation likelihood given the hypothesis x of which components are defective, $p(x)$ is the prior fault probability, and $p(o)$ is the probability that o is observed. However, the difficulty with this theoretically-sound formulation is that the hypothesis space is big — exponential in terms of the number of components in the system. This brings prohibitively high computation complexity when updating the posterior $p(x|o)$.

In this section, we propose a fast approximation approach to estimate component fault probabilities with-

t	p	z	g	b	df_1		df_2	
					su_1	lg_1	su_2	lg_2
1	ABCDE	f			ABCDE			
2	ABC	f			ABC			
3	ADE	f			ABC			
4	A	s	$\overline{A}$		BC	$\overline{A}$		
5	ADE	f	$\overline{A}$		BC	$\overline{A}$	DE	$\overline{A}$
6	AC	s	$\overline{AC}$		B	$\overline{AC}$	DE	$\overline{A}$
7	ADC	f	$\overline{AC}$	D	B	$\overline{AC}$		
8	ACE	s	$\overline{ACE}$	D	B	$\overline{AC}$		
9	B	s	$\overline{ACE}$,$\overline{B}$	D	[AB][BC]	$\overline{AC}$, $\overline{B}$		
10	AB	f	$\overline{ACE}$,$\overline{B}$	[AB],D				

Table 3: System with five components $C = \{A, B, C, D, E\}$ where $[AB]$ and D are faulted.

out handling the exponential complexity. We first consider an example:

- (ABC) fails
- (AE) fails
- (ABD) fails
- (CD) fails

The observations suggest that there are at least two faults in the system. The min-cardinality diagnosis is AC or AD. If all components have equal prior probability of being defective, our intuition is that A is the most likely component to be faulty because it has the highest count of occurence in failed observations (3 in this case). This makes sense in statistical inference: we can count the number of occurence of each component in failed observations, weighted by a propoer weight. The component with the highest count is the most likely to have fault.

Rather than computing the posterior with respect to the fault combination of hypothesis x, we seek to compute the probability that module m is having a fault given a failed observation o:

$$p(m = 1|o = 1) = \frac{p(m = 1)p(o = 1|m = 1)}{p(o = 1)}. \quad (4)$$

This formulation decouples component from each other in the sense that only the posterior marginal probabilities $p(m|o)$, rather than the joint posterior probabilities $p(x|o)$ are tracked. This is a much simpler computation, but incurs the loss in information. From the posterior marginals, one may not be able to fully reconstruct the joint probability. We can define the following probabilities:

- $p(m = 0) = r_m$ is the prior probability that component m is faulty;
- $p(o = 1)$ is the probability that o is observed.
- $p(o = 1|m = 1)$ is the probability that o is observed given that m has fault. Note that if m is not involved in o, we have $p(o = 1|m = 1) = p(o = 1)$.

Taking logarithm of the above, we have: Note that $\log p(m = 1)$ is a constant term, hence the only difference is the weight $c_m \stackrel{\triangle}{=} \log p_{o|m} - \log p_o$.

We now outline a simple algorithm: given an observation sequence $o^1, o^2, \cdots, o^T$, we compute the accumulative weight for each component m. For any failure observation o^t, we compute the weight c_m^t for each module m. If module m is not involved in the observation, the weight c_m^t is set to 0. This weight is then accumulated over all observations. The component with a high accumulative c_m is likely to be defective.

By summing up the total number of appearance weighted by $\log p_o$, the algorithm above is effectively computing $\sum_o \log p(m = 1|o)$. This is similar to, but not identical to computing $\log p(m = 1|o_1, o_2, \cdots, o_T)$. If o_i are mutually independent given $m = 1$, then the two are identical. Taking the most suspected module is similar to find the module with highest posterior $arg \max \log p(m = 1|o_1, \cdots o_T)$.

The computation of c_m is easy. It boils down to the computation of two terms:

- $p(o = 1)$ is the probability that a failure is observed. It is straightforward to show that $p(o = 1) = 1 - \prod_{k \in o}(1 - r_k q_k)$, where k is the number of modules involved in observation o, r_k is the prior probability that component k is defective, and q_k is the intermittency probability (i.e., the probability that component k malfunctions given that it is defective). Note that $r_k q_k$ is the probability that component k malfunctions. The derivation is straightforward: the output is good if and only if all modules involved do not malfunction. In the case of persistent faults, it is simply $p(o = 1) = 1 - \prod_{k \in o}(1 - r_k)$.
- $p(o = 1|m = 1)$ is the likelihood. If the fault is persistent, we have $p(o = 1|m = 1) = 1$ if module m is involved in o. On the other hand, if the fault is intermittent, $p(o = 1|m = 1)$ is more complicated, since the observation is not only dependent on module m, but also on other modules involved. We use R to denote the event that some modules (other than m) malfunctions, then we have $p(R = 1) = 1 - \prod_n (1 - r_n q_n)$. Since all the r's and q's are know, we can evaluate this term easily.

The modification from the qualitive diagnosis discussed earlier to statistical inference is relatively simple: rather than identify the bad modules with certainty, we identify the highly suspected modules.

7 QUALITATIVE ATTRIBUTE BASED DIAGNOSIS

Unlike conventional qualitative reasoning about quantities such as voltage, current, force, pressure, velocity, etc., we introduce reasoning over the attributes of

$$\log p(m=1|o=1) = \begin{cases} \log p_{o|m} - \log p_o + \log p(m=1) & \text{if } m \in o \\ \log p(m=1) & \text{if } m \notin o \end{cases} \quad (5)$$

$$\begin{aligned} p(o=1|m=1) &= \textstyle\sum_{R=0,1} p(o=1|m=1,R)p(R) \\ &= 1 \cdot p(R=1) + q_m \cdot p(R=0)) \\ &= (1 - \textstyle\prod_n (1 - r_n q_n)) + q_m \prod_n (1 - r_n q_n) \end{aligned} \quad (6)$$

the objects being manipulated by the system. For example, a sheet of paper has attributes including blank, ripped, dogeared and scuffed. In conventional model-based diagnosis, an abnormal output can be caused by any component upon which the output depended. This makes sense when reasoning about circuits or physical mechanisms. However, we can reason with far greater diagnostic precision. When technicians see a printed sheet when it should be blank, they know that it can only be caused by a printer within the machine and not by the paper tray. It is physically impossible for a paper tray to print on a sheet of paper. In model-based diagnosis terms, the conflict corresponding to an observed incorrect print only contains the printers within the machine.

A single observed sheet can lead to multiple conflicts. For example, a sheet might be misprinted and dogeared. The system thus may have two faults: a printer and an paper handling component.

These extensions make no fundamental change to the algorithms presented earlier. At each time, the corresponding algorithm of Section 6 takes as input the set of conflicts generated through the use of attributes, instead of the plan itself.

8 CONCLUSIONS

This paper is a first step towards an integrated qualitative diagnostic approach to systems which process material such as manufacturing lines and printers. It presents a novel algorithm for diagnosing multiple interaction faults which is far more memory efficient than the traditional model-based algorithms. The overall approach is similar to how technicans address troubleshooting.

REFERENCES

(Collins and Forbus, 1987) John W. Collins and Kenneth D. Forbus. Reasoning about fluids via molecular collections. In *AAAI*, pages 590–594, 1987.

(Darwiche, 2001) Adnan Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

(Davis, 1984) R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1):347–410, 1984.

(de Kleer and Williams, 1987) J. de Kleer and B. C. Williams. Diagnosing multiple faults. *aij*, 32(1):97–130, April 1987. Also in: *Readings in NonMonotonic Reasoning*, edited by Matthew L. Ginsberg, (Morgan Kaufmann, 1987), 280–297.

(de Kleer *et al.*, 1992) J. de Kleer, A. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

(de Kleer, 2007a) J. de Kleer. Diagnosing intermittent faults. In *18th International Workshop on Principles of Diagnosis*, pages 45–51, Nashville, USA, 2007.

(de Kleer, 2007b) J. de Kleer. Modeling when connections are the problem. In *Proc 20th IJCAI*, pages 311–317, Hyderabad, India, 2007.

(Dvorak and Kuipers, 1989) D. Dvorak and B. Kuipers. Model-based monitoring of dynamic systems. In *Proc. 11th IJCAI*, pages 1238–1243, Detroit, 1989.

(Eiter and Gottlob, 1995) Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.

(Fromherz *et al.*, 2003) M.P.J. Fromherz, D.G. Bobrow, and J. de Kleer. Model-based computing for design and control of reconfigurable systems. *The AI Magazine*, 24(4):120–130, 2003.

(Grastien *et al.*, 2007) Alban Grastien, Anbulagan, Jussi Rintanen, and Elena Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In *AAAI*, pages 305–310, 2007.

(Koren and Kohavi, 1977) Israel Koren and Zvi Kohavi. Diagnosis of intermittent faults in combinational networks. *IEEE Trans. Computers*, 26(11):1154–1158, 1977.

(Kuhn *et al.*, 2008) Lukas Kuhn, Bob Price, Johan de Kleer, Minh Do, and Rong Zhou. Pervasive diagnosis: Integration of active diagnosis into production plans. In *proceedings of AAAI*, Chicago, Illinois, USA, 2008.

(Muscettola *et al.*, 1998) Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–47, 1998.

(Reiter, 1987) R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96, 1987.

(Shearer *et al.*, 1971) J. L. Shearer, A. T. Murphy, and H. H. Richardson. *Introduction to System Dynamics*. Addison Wesley, Reading, MA, 1971.

(Williams and Nayak, 1996) B. C. Williams and P. P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI96)*, pages 971–978, 1996.