Online Model-Based Diagnosis of Production Systems

Lukas Kuhn, Johan de Kleer
Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 USA
{lkuhn,dekleer}@parc.com

Abstract

This paper extends model-based diagnosis (MBD) (Reiter 1987; de Kleer & Williams 1987) to systems which convert, move and process material. Examples of such systems are printers, refineries and food processing plants. Such plants present two challenges to model-based diagnosis: (1) the plant may process 100s-1000s of items per minute so 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 in a data structure of fixed size. We address the second challenge by introducing the notion of interaction fault which represents the situation where a set of components operating on the same object damage the object even though each component alone produces no 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.

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 such as in is characterized by system dynamics (Shearer, Murphy, & Richardson 1971). However, most real world systems transport and modify materials. 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 General Mills plant converts wheat and cardboard into boxes containing donut-shaped objects (Cheerios). Such systems need to reason about both the attributes of the stuff (e.g., voltage, current, pressure) and their properties (e.g., wrapped candy bar, unwrapped candy bar, partially assembled automobile).

Plants present two challenges to model-based diagnosis: (1) the plant processes 100s-1000s of items per minute so 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. This paper outlines an integrated approach to both challenges. First, we synopsize the results in a single fixed-size data structure. Necessarily some information will be lost and we rely on high throughput rates to "make up" for any information so lost. 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 which specifies some misbehavior has occured, but omitting details on exactly how.

We draw many of our examples from a prototype printer illustrated in Figure 1 ((Fromherz, Bobrow, & de Kleer 2003) provides more background).

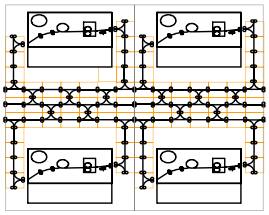


Figure 1: Model of PARC's prototype highly redundant printer. It consists of 4 printers (large rectangles). Sheets enters on the left and exit on the right.

Refineries, printers, manufacturing lines all run continuously. 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 & Forbus 1987). Although more difficult to analyze, continous operation has the advantage that it is possible to gather a great deal of observations quickly and cheaply.

The fact that objects are being processed by the system introduces a whole new set of fault types. For example, we will often see situations where component A operates correctly stand-alone, and component B operates correctly 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

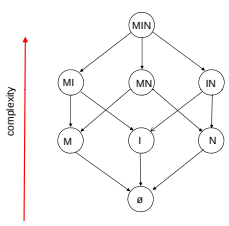
destroys the candy bar. We call such faults interaction faults: A and B are perfectly operational individually but will not work correctly if A and B both process the same candy bar. The classical model-based diagnosis approach would be to consider both components A and B as faulted, but that is not 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.

A technician reasons about a system at multiple levels of abstraction. A technician will make the simplest assumptions possible to diagnose a system and only when those assumptions yield a contradiction will he/she choose a more detailed model. We adopt the meta-diagnosis abstraction framework of (de Kleer 2007). In this approach, the meta-assumptions (of the modeling approach itself) are treated as assumptions in a model-based diagnosis engine. The system always picks one particular diagnosis as the current abstraction level.

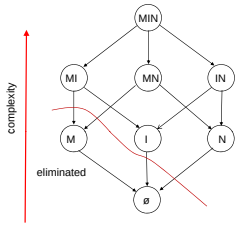
Example: Simplest Meta-Diagnosis Failing

For simplicity, we presume that if components are persistent faulty they will always manifest bad behavior. This assumption can also be a meta-assumption, but this makes the examples too complicated.

The three initial meta-assumptions we make are: (1) the system does not have multiple faults "M" (vs. single fault), (2) the fault is not intermittent "I" (vs. persistent), (3) the fault is not interactive "N." This corresponds to the bottom node of Figure 2(a).



(a) M indicates multiple faults; I indicates intermittent faults; N indicates interaction faults



(b) After the minimal conflict $AB_a(I) \vee AB_a(N)$.

Figure 2: Meta-Diagnosis lattice

Consider only the components A, B, C. 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

A plan $p_i = [c_1, c_2, \ldots, c_n]$ is a sequence of components involved in an execution. 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:

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

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:

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

Figure 2(b) 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 $[AB]$. An interaction fault is one in which both components might individually be working correctly, but produce faulty behavior when combined. We use $[...]$ to indicate the interaction fault which occurs only when all of the components operate on the same object. Plan 1 is the only plan in which A and B co-occur, therefore the interaction fault explains all symptoms.

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 & Gottlob 1995).

Figure 3 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.

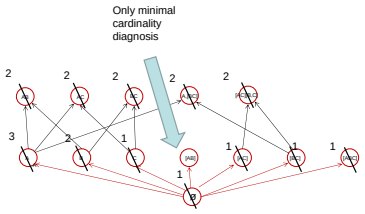


Figure 3: Fragment of diagnosis lattice for the simple 3 component system A, B, C. Includes multiple and interaction faults. The numbers indicate which plan eliminates that diagnosis.

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.

Diagnostic Algorithm

In this section we present a new diagnostic algorithm which differs from classical model-based diagnostic algorithms in significant ways. The new diagnostic algorithm maintains 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 a 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 algorithm we describe may take more observations to pinpoint the true fault(s), but it will never miss faults.

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.

A **system** Sys is 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.

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.

Algorithm 1: Multiple (Interaction) Faults Algorithm with Memory

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

Algorithm 1 executes Procedure 2 (or for the interaction fault case Procedure 4) 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.

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 1 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 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 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 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)

Function `multipleFaults` (*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;
```

in the unknown set. Plan 8 (ACE) succeeds, thus ACE is exonerated.

Multiple Interaction Faults Case

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 defined in the multiple fault case.

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.

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

Consider the following example. Let Sys be a simple system with five components $C = \{A, B, C, D, E\}$. Suppose component B and D are faulted. In Table 2 we show walk through the example.

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

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 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.

Function *multiInteractFaults(plan p_j, z_j)* descripes the algorithm for multiple interaction faults in more detail.

Functioncandidates and minimalCandidates

```
minimalCandidates(Set<Comps> C, P̄(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
candidates(Set<Comps> C, P̄(Set<Comps>) PC)
Beginn
    CA = P(C);
    foreach pc_i : PC do
        CA = CA − pc_i;
    return CA;
Ende
```

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, J. W., and Forbus, K. D. 1987. Reasoning about fluids via molecular collections. In *AAAI*, 590–594.

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

de Kleer, J. 2007. Modeling when connections are the problem. In *Proc 20th IJCAI*, 311–317.

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *Journal of the ACM* 42(1):3–42.

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

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

Shearer, J. L.; Murphy, A. T.; and Richardson, H. H. 1971. *Introduction to System Dynamics*. Reading, MA: Addison Wesley.

Function multiInteractFaults (*plan p_j, obs 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 ⊔ overline(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 ⊔ overline(E(c) ∩ p_j);
                                su_i = MCA_j;
                            else
                                return false;
                        else
                            return false;
else
    g = g ⊔ overline(p_j);
    foreach df_i : DF do
        lg_i = lg_i ⊔ overline(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;
```