

Generating Diagnoses from Conflict Sets

Rolf Haenni

Institute of Informatics
University of Fribourg
Switzerland
rolf.haenni@unifr.ch

Abstract

Many techniques of finding possible diagnoses of faulty technical systems involve two sequential steps. First compute the collection of all minimal conflict sets, then transform the conflict sets into diagnoses. This paper addresses the second step of this procedure. We assume that the conflict sets are known and we present an efficient method of transforming conflict sets into diagnoses. The method is developed in a more general framework for the corresponding problem of computing hypergraph inversions.¹

Introduction

It can happen that technical systems composed of several components do not operate as they were designed to be operating. This discrepancy between the expected and observed behavior of the system is due to the malfunctioning of one or more components. The diagnostic problem is to identify those faulty components responsible for the malfunctioning of the system. Of course, when the faulty components are identified, they must be replaced by non-faulty components in order to obtain a system that is working correctly.

Many different theories and models to handle the diagnostic problem have been developed. One important approach is based on Reiter's theory of diagnosis from first principles (Reiter 1987). Other substantial contributions to this approach are those of (Davis 1984), de Kleer et al. (de Kleer 1976; de Kleer & Williams 1987; de Kleer, Mackworth, & Reiter 1992), Genesereth (Genesereth 1984), Reggia et al. (Reggia, Nau, & Wang 1983; 1985), and Kohlas et al. (Kohlas *et al.* 1996).

Reiter's approach can be summarized as follows: a **system** is a triple (CMP, SD, OBS) ; CMP is a finite set of **system components**; SD is the **system description**, a finite set of first-order sentences; OBS is the **observation**, another finite set of first-order sentences describing the observed input-output values; a **diagnosis** is a minimal set $D \subseteq CMP$ of components, such that "the assumption that each of these components is

faulty, together with the assumption that all other components are behaving correctly, is consistent with the system description and the observation" (Reiter 1987). The diagnostic problem consists then in finding all diagnoses for a system (CMP, SD, OBS) . The classical approach of finding diagnoses involves two sequential steps: (1) compute the collection of all minimal conflict sets; (2) transform the conflict sets into diagnoses. A **minimal conflict set** is a minimal set $C \subseteq CMP$ of components, such that the assumption that each of these components is behaving correctly is inconsistent with the system description and the observation.

In this paper we address step (2) in the classical diagnostic procedure: we assume that all minimal conflict sets are known and we present an efficient method of transforming conflict sets into diagnoses. Reiter uses the notion of **minimal hitting sets** and shows that diagnoses are minimal hitting sets of the conflict sets. Hitting sets can be computed by so-called **HS-trees**. The problem with complete HS-trees is that the size of the tree grows exponentially with the size of the incoming collection of sets. Minimal hitting sets are obtained more efficiently by **pruned HS-tree**. The idea is that subtrees producing only non-minimal hitting sets are pruned off. However, the construction of pruned HS-trees is difficult to organize such that no unnecessary results are generated. Often, unnecessary subtrees are detected only after the entire subtree has been generated. The problem is that Reiter's method does not specify the order of the incoming collection of sets. If the construction of the tree starts with an unfavorable set, then the method is always producing some non-minimal results regardless whether the tree is constructed breadth-first or depth-first.

The method presented in this paper is developed in a more general framework for the corresponding problem of computing **hypergraph inversions**. The collection of minimal conflict sets is considered as a hypergraph on C . By sequentially eliminating the leaves of the hypergraph it is guaranteed that no unnecessary results are generated. However, in the worst case the size of the resulting hypergraph inversion grows exponentially. Then the method presented in this paper can easily be adapted in order to produce only the most important

¹Copyright 1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

results. Furthermore, the method can be extended to solve the problem of transforming CNFs into DNFs and inversely. In the domain of model-based diagnostics this is important when the **minimal diagnosis hypothesis** does not hold (de Kleer, Mackworth, & Reiter 1992).

Note that the problem of computing diagnoses from minimal conflict sets is also related to the well-known **set covering problem** (Cormen, Leiserson, & Rivest 1989). In fact, a diagnosis is a minimal **set cover** of the collection of conflict sets. The set covering problem consists only in finding one solution of minimal size. The algorithms for solving the set covering problem are therefore not useful if all diagnoses have to be found.

Hypergraph Inversion

Let $N = \{n_1, \dots, n_r\}$ be a finite set of **nodes** n_i . A **hypergraph** $H = \{e_1, \dots, e_m\}$ on N is a set of subsets $e_i \subseteq N$. In other words $H \subseteq 2^N$. The elements of H are called **hyperedges**. A hyperedge $e_i \in H$ is called **minimal** in H , if there is no other hyperedge $e_j \in H$, $i \neq j$, such that $e_j \subseteq e_i$. A hypergraph is called **minimal**, if every hyperedge $e_i \in H$ is minimal. A hypergraph is called **simple**, if H has only one hyperedge. A node $n_i \in N$ is called **leaf** of H , if n_i appears only in one hyperedge.

Hypergraphs can easily be represented graphically. For example, let

$$H = \{\{1, 2, 3\}, \{1, 3, 5\}, \{1, 6\}, \{2, 4\}, \{2, 3, 5\}, \{2, 4, 5\}\}$$

be a hypergraph for $N = \{1, 2, 3, 4, 5, 6\}$. This corresponds to Example 4.7 in (Reiter 1987). Figure 1 shows a graphical representation of H .

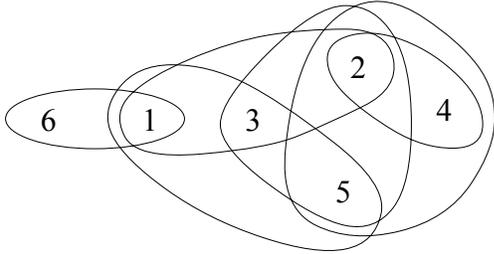


Figure 1: A hypergraph representation.

Note that H is not minimal since the hyperedge $\{2, 4, 5\}$ is not minimal. A corresponding minimal hypergraph μH can be obtained by dropping all non-minimal hyperedges. In this way $\{2, 4, 5\}$ can be dropped and we get $\mu H = \{\{1, 2, 3\}, \{1, 3, 5\}, \{1, 6\}, \{2, 4\}, \{2, 3, 5\}\}$. Furthermore, note that 6 is a leaf of H , whereas 6 and 4 are leaves of μH .

A set of nodes $e \subseteq N$ is called **connecting hyperedge** relative to H , if $e \cap e_i \neq \emptyset$ for all $e_i \in H$. For example, $\{1, 2\}$ is a connecting hyperedge relative to the hypergraph in Fig. 1. The set of minimal connecting hyperedges relative to H is called **hypergraph inversion**

of H , denoted by H^{-1} . Note that $H^{-1} = (\mu H)^{-1}$. For minimal hypergraphs we have the following theorem:

Theorem 1 *If H is a minimal hypergraph, then*

$$(H^{-1})^{-1} = H. \quad (1)$$

Note that if the elements of H are the (minimal) conflict sets of a faulty technical system, then the elements of H^{-1} are the diagnoses. Therefore, the problem of finding diagnoses from conflict sets corresponds to the problem of computing hypergraph inversions.

Hypergraph inversions can be computed by sequentially eliminating leaves. Let $n \in N$ be a leaf of H and e_i the corresponding hyperedge containing n . Then H can be decomposed into two other hypergraphs H_{+n} and H_{-n} in which n does not appear any more:

$$H_{+n} := \mu\{e_j - e_i : e_j \in H, i \neq j\}, \quad (2)$$

$$H_{-n} := \mu\{e_j - n : e_j \in H\}. \quad (3)$$

Again, consider the hypergraph H of Fig. 1. Eliminating the leaf 6 leads to $H_{+6} = \{\{2, 3\}, \{2, 4\}, \{3, 5\}\}$ and $H_{-6} = \{\{1\}, \{2, 4\}, \{2, 3, 5\}\}$.

Note that in some cases this procedure may produce $H_{+n} = \emptyset$ or $H_{-n} = \{\emptyset\}$. For that purpose let's define $\emptyset^{-1} = \{\emptyset\}$ and $\{\emptyset\}^{-1} = \emptyset$. The hypergraph inversion H^{-1} of H can then be obtained according to the following theorem:

Theorem 2 *If n is a leaf in a hypergraph H , H_{+n} and H_{-n} as defined in (2) and (3), then*

$$H^{-1} = \{e \cup \{n\} : e \in (H_{+n})^{-1}\} \cup (H_{-n})^{-1}. \quad (4)$$

Theorem 2 describes a recursive method of computing hypergraph inversions by sequentially eliminating leaves. The idea is that the minimal connecting hyperedges containing n are determined using H_{+n} , while the minimal connecting hyperedges not containing n are obtained from H_{-n} . Note that the solutions generated by H_{+n} and H_{-n} are disjoint in the sense that the right-hand side of (4) is automatically a minimal hypergraph. This is the crucial point of this approach: no unnecessary solutions are generated.

This recursive process can be simplified if instead of $\emptyset^{-1} = \{\emptyset\}$ and $\{\emptyset\}^{-1} = \emptyset$ we use another condition for simple hypergraphs: let e be the only hyperedge of a simple hypergraph $H = \{e\}$, then

$$H^{-1} = \{\{n\} : n \in e\}. \quad (5)$$

The recursive process of computing the hypergraph inversion of above example is depicted in Fig. 2.

The tree shows how the original hypergraph is consecutively decomposed until simple hypergraphs are produced. Finally, the hypergraph inversion is obtained by (4) and (5). For our example we get the following solution:

$$H^{-1} = \{\{1, 2\}, \{1, 3, 4\}, \{1, 4, 5\}, \{2, 3, 6\}, \{2, 5, 6\}, \{3, 4, 6\}\}.$$

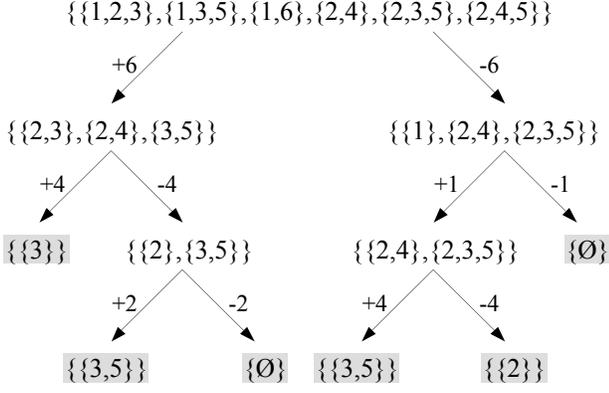


Figure 2: Recursive process for computing H^{-1} .

The attentive reader may have recognized that this approach of eliminating leaves is only working if every hypergraph produced during the recursion has at least one leaf. A hypergraph with at least one leaf is called **regular**. Figure 3 shows a non-regular minimal hypergraph. In such a case the above decomposition is not applicable.

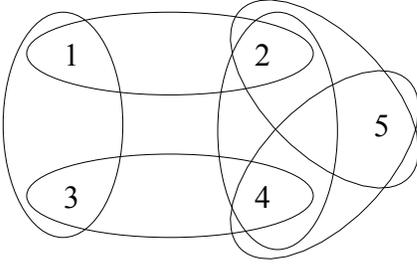


Figure 3: A non-regular hypergraph.

Consider an arbitrary node n in a (regular or non-regular) hypergraph H . If $E_n \subseteq H$ is the set of hyperedges containing n , then a more general decomposition can be defined:

$$H_{+n} := \mu\{e_j - \cap E_n : e_j \in H - E_n\}, \quad (6)$$

$$H_{-n} := \mu\{e_j - n : e_j \in H\}. \quad (7)$$

This way of decomposing H replaces (2) and (3) since it includes the case of n being a leaf by $E_n = \{e_i\}$. For a non-regular hypergraph H we can now obtain the hypergraph inversion H^{-1} as follows:

Theorem 3 *If n is a node in a hypergraph H , H_{+n} and H_{-n} as defined in (6) and (7), then*

$$H^{-1} = \mu(\{e \cup \{n\} : e \in (H_{+n})^{-1}\} \cup (H_{-n})^{-1}). \quad (8)$$

The only difference between Theorem 2 and Theorem 3 is the necessity of minimizing the result in Theorem 3. Note that the operation μ can be optimized for this particular case. If we define $H_1 := \{e \cup \{n\} : e \in (H_{+n})^{-1}\}$ and $H_2 := (H_{-n})^{-1}$, then the problem in Theorem 3 is to compute $\mu(H_1 \cup H_2)$. Clearly, H_1 and

H_2 are minimal. Furthermore, every hyperedge in H_2 is minimal in H_1 . Therefore, we have only to check if H_1 contains hyperedges which are not minimal in H_2 .

Let $n = 1$ be the node to be eliminated from the example in Fig. 3. The hypergraph can then be decomposed as follows:

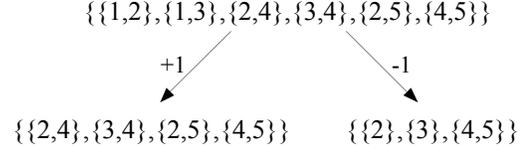


Figure 4: Decomposing a non-regular hypergraph.

Note that both H_{+n} and H_{-n} are regular. To obtain the hypergraph inversions $(H_{+1})^{-1}$ and $(H_{-1})^{-1}$ we can therefore use Theorem 2 for regular hypergraphs to get the following results:

$$(H_{+1})^{-1} = \{\{2,3,5\}, \{2,4\}, \{4,5\}\},$$

$$(H_{-1})^{-1} = \{\{2,3,4\}, \{2,3,5\}\}.$$

Finally, we can use Theorem 3 to compute the hypergraph inversion for the example in Fig. 3:

$$H^{-1} = \{\{1,2,4\}, \{1,4,5\}, \{2,3,4\}, \{2,3,5\}\}.$$

Note that $\{2,3,5\} \in (H_{+1})^{-1}$ is not relevant for the final result because the same hyperedge is already contained in $(H_{-1})^{-1}$. Therefore, this example produces only one unnecessary result, since during the computation of $(H_{+1})^{-1}$ and $(H_{-1})^{-1}$ only regular hypergraphs arise.

In the domain of model-based diagnostics, i.e. when H represents the minimal conflict sets for a system (CMP, SD, OBS), non-regular hypergraphs are rare. Typically, non-regular hypergraphs are only observed in less than one percent of the recursive calls. Similar results are observed in other domains.

This method of computing hypergraph inversions can be described by the following recursive algorithm:

Algorithm 1 (*Hypergraph Inversion*)
- Input: hypergraph H
- Output: hypergraph inversion H^{-1}
BEGIN
IF H is simple
THEN RETURN $\{\{n\} : n \in e, e \in H\}$;
ELSE
IF H is regular
THEN SET $n :=$ leaf of H ;
ELSE SET $n :=$ node of H ;
ENDIF;
SET H_+ according to (6);
SET H_- according to (7);
COMPUTE H_+^{-1} and H_-^{-1} recursively;
SET $H_* := \{e \cup \{n\} : e \in H_+^{-1}\} \cup H_-^{-1}$;
IF H is regular
THEN RETURN H_* ;

```

ELSE RETURN  $\mu H_*$ ;
ENDIF;
ENDIF;
END.

```

A leaf can easily be selected from a hypergraph. The following algorithm determines the set L of all leaves of a hypergraph H . Note that H is regular if and only if $L \neq \emptyset$.

Algorithm 2 (*Computing Leaves*)

- *Input:* hypergraph H
- *Output:* set L of all leaves of H

```

BEGIN
SET  $L := \emptyset$ ;
SET  $N := \emptyset$ ;
LOOP FOR  $e$  in  $H$  DO
SET  $L := (L - e) \cup (e - N)$ ;
SET  $N := N \cup e$ ;
ENDLOOP;
RETURN  $L$ ;
END.

```

Actually, every leaf $n \in L$ can be selected for being eliminated from a regular hypergraph H . However, let's remarks that Theorem 2 and Alg. 1 can easily be adapted such that leaves from the same hyperedge can be eliminated simultaneously. Then it is recommendable to select the leaves of the hyperedge with a maximal number of leaves. This reduces the necessary number of calling Alg. 2 recursively.

Similarly, every node $n \in N$ can be selected for being eliminated from a non-regular hypergraph H . Here it is recommendable to select a node n such that the size of E_n is maximal. This ensures that the size of $H - E_n$ is minimal and therefore that H_{+n} in (6) is as small as possible. This reduces the number of unnecessary results generated by $(H_{+n})^{-1}$.

Let s be the number of resulting hyperedges in H^{-1} and let's assume that computing H^{-1} produces only regular hypergraphs. If we neglect the time needed for selecting a leaf in a hypergraph by Alg. 2 as well as the cost of decomposing the hypergraph according to (6) and (7), then the time needed for computing H^{-1} by Alg. 1 grows linearly with s . Of course, s itself may grow exponentially with the number of hyperedges in H , for example in the case when H consists only of leaves (worst case). Then, the complexity lies within the nature of the problem and Alg. 1 is only feasible, if the result H^{-1} consists of a relatively small number s of connecting hyperedges.

Approximating Hypergraph Inversion

The problem of computing hypergraph inversions is that the number s of resulting connecting hyperedges in H^{-1} grows exponentially with the number of hyperedges in H . However, it may often not be necessary to know the complete set H^{-1} . In the domain of model-based diagnostics, for example, only short diagnoses are interesting. The idea of more and less important connecting hyperedges will be developed in this section. In

this sense the algorithm of Section will be adapted in order to approximate hypergraph inversions.

Generating Short Hyperedges

The number of nodes in a hyperedge e is called **size** of e , denoted by $|e|$. If $N = \{n_1, \dots, n_r\}$ is the set of nodes, then $E_k = \{e \in 2^N : |e| \leq k\}$ denotes the set of all hyperedges in N with a size smaller than k . Let $H = \{e_1, \dots, e_m\}$ be a hypergraph on N . The **hypergraph inversion of order k** can then be defined by

$$H^{-1}(k) = H^{-1} \cap E_k. \quad (9)$$

Note that we have $H^{-1}(0) = \emptyset$ for an arbitrary $H \neq \emptyset$. Furthermore, if m is the number of hyperedges in H and $n \geq m$, then $H^{-1}(n) = H^{-1}$.

Consider the example of Fig. 1. The hypergraph inversions of order k , $k = 0, 1, \dots$ are then

$$\begin{aligned}
H^{-1}(0) &= \emptyset, \\
H^{-1}(1) &= \emptyset, \\
H^{-1}(2) &= \{\{1, 2\}\}, \\
H^{-1}(3) &= \{\{1, 2\}, \{1, 3, 4\}, \{1, 4, 5\}, \{2, 3, 6\}, \\
&\quad \{2, 5, 6\}, \{3, 4, 6\}\}, \\
H^{-1}(4) &= \{\{1, 2\}, \{1, 3, 4\}, \{1, 4, 5\}, \{2, 3, 6\}, \\
&\quad \{2, 5, 6\}, \{3, 4, 6\}\}, \\
&\vdots
\end{aligned}$$

The algorithm of Section can now be adapted for the computation of $H^{-1}(k)$. The idea is that at each step of the recursion the computation of $(H_{+n})^{-1}$ starts with $k-1$. An additional condition $k = 0$ stops the recursion.

Algorithm 3 (*Hypergraph Inversion of Order k*)

- *Input:* hypergraph H , order k
- *Output:* hypergraph inversion of order k

```

BEGIN
IF  $k = 0$ 
THEN RETURN  $\emptyset$ ;
ELSE
IF  $H$  is simple
THEN RETURN  $\{\{n\} : n \in e, e \in H\}$ ;
ELSE
IF  $H$  is regular
THEN SET  $n := \text{leaf of } H$ ;
ELSE SET  $n := \text{node of } H$ ;
ENDIF;
SET  $H_+$  according to (6);
SET  $H_-$  according to (7);
COMPUTE  $H_+^{-1}(k-1)$  and  $H_-^{-1}(k)$  recursively;
SET  $H_* := \{e \cup \{n\} : e \in H_+^{-1}(k-1)\} \cup H_-^{-1}(k)$ ;
IF  $H$  is regular
THEN RETURN  $H_*$ ;
ELSE RETURN  $\mu H_*$ ;
ENDIF;
ENDIF;
ENDIF;
END.

```

With this algorithm it is now possible to compute the shortest connecting hyperedges even for big hypergraphs. However, the worst case (i.e. if H only consists of leaves) is still not feasible because the hyperedges of H^{-1} have all the same size. In such cases the method of the following subsection may help to obtain the most relevant results.

Generating Relevant Hyperedges

In this subsection we develop the idea that some nodes of the hypergraph are more important than others. Therefore, connecting hyperedges consisting of more important nodes are more relevant than hyperedges consisting of less important nodes. Suppose that for each node $n_i \in N$ a corresponding **relevance factor** $0 \leq p(n_i) \leq 1$ is given. In the sequel, relevance factors are treated as probabilities. The **relevance** $p(e)$ of a hyperedge is then given by the product of the relevance factors of the nodes in e :

$$p(e) = \prod \{p(n_i) : n_i \in e\}. \quad (10)$$

Note that in most cases the shortest hyperedges are also the most relevant hyperedges. If p is a number between 0 and 1, then $E_p = \{e \in 2^N : p(e) \geq p\}$ denotes the set of all hyperedges in N with a relevance higher than p . Let $H = \{e_1, \dots, e_m\}$ be a hypergraph on N . The **hypergraph inversion of relevance** p can then be defined by

$$H^{-1}(p) = H^{-1} \cap E_p. \quad (11)$$

Again, consider the example of Fig. 1. Suppose that the following relevance factors are given: $p(1) = 0.9$, $p(2) = 0.8$, $p(3) = 0.7$, $p(4) = 0.6$, $p(5) = 0.5$, and $p(6) = 0.4$. The relevance of the connecting hyperedges in H^{-1} is then given by the following values:

$$\begin{aligned} p(\{1, 2\}) &= 0.72, & p(\{1, 3, 4\}) &= 0.378, \\ p(\{1, 4, 5\}) &= 0.27, & p(\{2, 3, 6\}) &= 0.224, \\ p(\{3, 4, 6\}) &= 0.168, & p(\{2, 5, 6\}) &= 0.16. \end{aligned}$$

For example, if we fix $p = 0.25$, then the hypergraph inversion of relevance 0.25 is

$$H^{-1}(0.25) = \{\{1, 2\}, \{1, 3, 4\}, \{1, 4, 5\}\}.$$

Again, the algorithm of Section can easily be adapted for the computation of $H^{-1}(p)$. The idea now is that at each step of the recursion the computation of $(H_{+n})^{-1}$ starts with $\frac{p}{p(n)}$. An additional condition $p > 1$ stops the recursion.

Algorithm 4 (Hypergraph Inversion of Relevance p)

- Input: hypergraph H , relevance p
 - Output: hypergraph inversion of relevance p
 BEGIN

```

IF  $p > 1$ 
THEN RETURN  $\emptyset$ ;
ELSE
  IF  $H$  is simple
  THEN RETURN  $\{\{n\} : n \in e, e \in H\}$ ;
  ELSE

```

```

IF  $H$  is regular
  THEN SET  $n := \text{leaf of } H$ ;
  ELSE SET  $n := \text{node of } H$ ;
ENDIF;
SET  $H_+$  according to (6);
SET  $H_-$  according to (7);
COMPUTE  $H_+^{-1}(\frac{p}{p(n)})$  and  $H_-^{-1}(p)$  recursively;
SET  $H_* := \{e \cup \{n\} : e \in H_+^{-1}(k-1)\} \cup H_-^{-1}(k)$ ;
IF  $H$  is regular
  THEN RETURN  $H_*$ ;
  ELSE RETURN  $\mu H_*$ ;
ENDIF;
ENDIF;
ENDIF;
END.

```

With this algorithm it is now possible to compute the most relevant connecting hyperedges even in the worst case when H only consists of leaves. The only case in which the algorithm is still not able to produce any result is when H consists only of leaves and when all the nodes $n_i \in N$ have the same relevance factors (the worst case of the worst case).

Acknowledgments

Research supported by grant No.2100-042927.95 of the Swiss National Foundation for Research.

References

- Cormen, T.; Leiserson, C.; and Rivest, R. 1989. *Introduction to Algorithms*. MIT Press.
- Davis, R. 1984. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence* 24:347-410.
- de Kleer, J., and Williams, B. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32:97-130.
- de Kleer, J.; Mackworth, A. K.; and Reiter, R. 1992. Characterizing diagnosis and systems. *Artificial Intelligence* 56:197-222.
- de Kleer, J. 1976. *Local Methods for Localizing Faults in Electronical Circuits*. MIT AI Memo 394, MIT Cambridge, MA.
- Genesereth, M. 1984. The use of design description in automated diagnosis. *Artificial Intelligence* 24:411-436.
- Kohlas, J.; Monney, P.; Anrig, B.; and Haenni, R. 1996. Model-based diagnostics and probabilistic assumption-based reasoning. Technical Report 96-09, University of Fribourg, Institute of Informatics.
- Reggia, J.; Nau, D.; and Wang, Y. 1983. Diagnostic expert systems based on set covering model. *Int. J. Man-Machine Stud.* 19:437-460.
- Reggia, J.; Nau, D.; and Wang, Y. 1985. A formal model of diagnostic inference: 1. problem formulation and decomposition. *Inf. Sci.* 37:227-256.
- Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57-95.