Controlling the Complexity in Model-Based Diagnosis*

Igor Mozetič
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
igor@ai.univie.ac.at

Christian Holzbaur
Austrian Research Institute for Artificial Intelligence, and
Department of Medical Cybernetics and Artificial Intelligence
University of Vienna
Freyung 6, A-1010 Vienna, Austria
christian@ai.univie.ac.at

Abstract

We present IDA — an *I*ncremental *D*iagnostic *A*lgorithm which computes minimal diagnoses from diagnoses, and not from conflicts. As a consequence of this, and by using different models, one can control the computational complexity. In particular, we show that by using a model of the normal behavior, the worst-case complexity of the algorithm to compute the $k+1$-st minimal diagnosis is $O(n^{2k})$, where n is the number of components. On the practical side, an experimental evaluation indicates that the algorithm can efficiently diagnose devices consisting of a few thousand components. We propose to use a hierarchy of models: first a structural model to compute all minimal diagnoses, then a normal behavior model to find the additional diagnoses if needed, and only then a fault model for their verification. IDA separates model interpretation from the search for minimal diagnoses in the sense that the model interpreter is replaceable. In particular, we show that in some domains it is advantageous to use the Constraint Logic Programming system CLP($\mathcal{B}$) instead of a logic programming system like Prolog.

*To appear in *Annals of Mathematics and Artificial Intelligence*, 1993. This is an extended version of the paper by Igor Mozetič "A polynomial-time algorithm for model-based diagnosis" which appears in the *Proc. European Conf. on Artificial Intelligence, ECAI-92* (B. Neumann, Ed.), pp. 729-733, John Wiley & Sons, 1992.

1 Introduction

Model-based diagnosis is the activity of locating malfunctioning components of a system solely on the basis of its structure and behavior. There are two prevailing approaches, consistency-based and abductive [Poole, 1989], which differ in the representation of knowledge about the normality and faults, and in how diagnoses are defined and computed. Recently, a number of negative results have been reported about the complexity of model-based diagnosis. In particular, in the consistency-based approach, finding the next minimal diagnosis with a 'weak' fault model (i.e., a model of a normal behavior) is an NP-complete problem [Friedrich *et al.*, 1990]. With a 'strong' fault model (i.e., a model of different faults) even computing the first (non-minimal) diagnosis is NP-complete. Similar results were shown in the context of abductive diagnosis [Bylander *et al.*, 1989].

The goal of this paper is to present IDA — an *I*ncremental *D*iagnostic *A*lgorithm which has polynomial worst-case time complexity on one hand, and, on the other hand, can efficiently diagnose large-scale devices. We identify two potential sources of exponential complexity: the search through the space of potential diagnoses, represented by a lattice, and the type of the fault model used. We define the TP function (theorem prover), originally proposed by Reiter [Reiter, 1987], which clearly separates the model interpretation from the lattice search. The TP function does not require ATMS-style dependency recording and can be easily realized by a logic programming system like Prolog. This also avoids incomplete constraint propagation which occurs in most ATMS-based systems [de Kleer and Williams, 1987]. Furthermore, without any change to the diagnostic algorithm, TP can be realized by different instances of the Constraint Logic Programming (CLP) scheme [Jaffar and Lassez, 1987], depending on the domain of application.

In section 2 we give a new characterization of models, diagnoses and conflicts, and show how to represent different types of models by logic programs. This is illustrated by the frequently used binary adder example; in section 4 the example is expanded and experimental results are compared to de Kleer's HTMS-based system [de Kleer, 1991].

The basic algorithm which computes all minimal diagnoses is described in section 3. In contrast to most consistency-based approaches where minimal diagnoses are computed from conflicts (e.g. [Reiter, 1987, de Kleer and Williams, 1987]), our algorithm computes minimal diagnoses directly from diagnoses (direct computation of diagnoses was also described in [Friedrich and Nejdl, 1990]). Conflicts are computed as a side effect, and are used to prune the search space. The algorithm is incremental in the sense that it can compute the $k+1$-st minimal diagnosis from the previous k diagnoses.

In section 4 we show that the IDA algorithm computes the $k+1$-st minimal diagnosis in time $O(n^{2k})$, where n is the number of the model components, provided the TP function requires constant time. In an experimental evaluation on a large device consisting of 5000 components, the algorithm computed the first 15 diagnoses in one minute of CPU time. Next we address the complexity of the TP function with respect to three classes of models: structural, weak, and strong fault models.

A *structural model* specifies just how components are interconnected. If there are f faulty outputs from the model then *all* minimal diagnoses are found in polynomial time $O(n^f)$. A *weak fault model* specifies connections and normal behavior of the model components. The algorithm finds the $k+1$-st diagnosis in $O(n^{2k})$ time. In addition, if the diagnoses computed by the structural model are verified first, then *all* minimal diagnoses of cardinality $\leq f$ are found in polynomial time. A *strong fault model* specifies all the possible ways in which a component can fail. Since even computing the first (non-minimal) diagnosis is NP-complete, we use the strong fault model just to verify the minimal diagnoses computed by the weak fault model. Minimal diagnoses are typically of low cardinality, and their verification is usually tractable. If no previously computed minimal diagnosis is admitted by the strong fault model, we can use the weak fault model to compute the next, $k+1$-st minimal diagnosis. An incremental application of the algorithm therefore guarantees a smooth degradation of performance.

Another advantage of an explicit TP function is that one can replace the underlying theorem prover without changing the diagnostic algorithm. In section 5 we show the impacts of realizing the TP function by the Constraint Logic Programming system CLP($\mathcal{B}$) where the standard computational domain (Herbrand universe) is extended by boolean expressions.

2 Modeling and diagnosing with logic programs

Model-based reasoning about a system requires an explicit representation (a model) of the system's components and their connections. Most diagnostic systems represent models in terms of constraints coupled with an ATMS [de Kleer and Williams, 1987, de Kleer and Williams, 1989], or as a set of propositions in first-order logic [Reiter, 1987]. In contrast, we represent models by *logic programs* [Lloyd, 1987] and by *constraint logic programs* [Jaffar and Lassez, 1987, Cohen, 1990]. Similar representation was proposed in [Saraswat *et al.*, 1990], but the origin goes back to the KARDIO model [Bratko *et al.*, 1989]. Definitions of basic concepts typically follow [Reiter, 1987] — we give an alternative, *relational* characterization, suitable for model representation by (constraint) logic programs.

Definition. A *model* of a system is a triple $\langle SD, COMPS, OBS \rangle$ where:

1. *SD*, the system description, is a (constraint) logic program with a distinguished binary predicate $m(COMPS, OBS)$ which represents a relation between the state of the system and the observations.

2. *COMPS*, states of the system components, is an n-tuple $\langle S_1, \ldots, S_n \rangle$ where n is the number of components, and variables S_i denote states (normal and abnormal) of components.

3. *OBS*, observations, is an m-tuple $\langle In_1, \ldots, In_i, Out_{i+1}, \ldots, Out_m \rangle$ where *In* and *Out* denote inputs and outputs of the model, respectively.

In a logic program, n-tuples are represented by terms of arity n. Variables start with capitals and are implicitly universally quantified in front of a clause, and constants start with lower-case letters. In *SD*, definitions, and algorithms, we refer to a distinguished constant *ok* to denote that the state S_i of the component i is normal. This corresponds to the statement $\neg ab(S_i)$ used in the consistency-based approach.

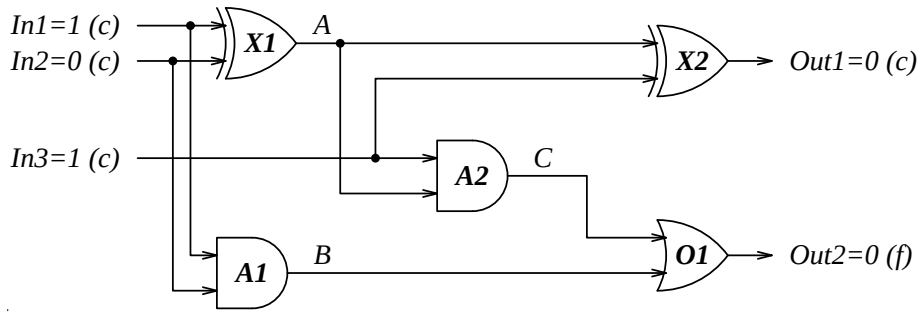

Figure 1: A binary adder, and an observation $\langle 1,0,1,\ 0,0\rangle$; c denotes a correct and f a faulty output.

Example (binary adder [Genesereth, 1984], Figure 1). The distinguished binary predicate m is *adder*, *COMPS* is a five-tuple $\langle X1,X2,A1,A2,O1\rangle$, and *OBS* is a five-tuple $\langle In1,In2,In3,\ Out1,Out2\rangle$. *SD* consists of the following clause which specifies the structure of the adder, and of additional clauses which define behavior of the components:

$adder(\ \langle X1,X2,A1,A2,O1\rangle,\ \langle In1,In2,In3,Out1,Out2\rangle)\ \leftarrow$
$xorg(\ X1,\ In1,\ In2,\ A\ ),$
$xorg(\ X2,\ In3,\ A,\ Out1\ ),$
$andg(\ A1,\ In1,\ In2,\ B\ ),$
$andg(\ A2,\ In3,\ A,\ C\ ),$
$org(\ O1,\ B,\ C,\ Out2\ ).$

Connections between components are represented by shared variables. Specification of the behavior depends on the type of the fault model available: structural, weak, exoneration, or strong. For illustration of different alternatives we define the behavior of an OR gate (*org*).

A **structural** model specifies the most general condition about the propagation of faults [Bakker *et al.*, 1989]. Only if all inputs to a component are correct (c) and the component is normal (ok) then the output is correct. In other words, given the observation that all inputs are correct (c) and the output is faulty (f) then the component is abnormal (ab). Otherwise, nothing can be concluded about the component state. A more compact encoding then the one presented here, namely by CLP($\mathcal{B}$), is described in section 5.

$org(\ ok,\ c,\ c,\ c\ ).$
$org(\ ok,\ c,\ f,\ \_\ ).$
$org(\ ok,\ f,\ \_,\ \_\ ).$
$org(\ ab,\ \_,\ \_,\ \_\ ).$

A **weak** fault model defines just normal behavior of components (state *ok*), abnormal behavior (state *ab*) is unconstrained:

org(ok, X, Y, Z) $\leftarrow$ *or(X, Y, Z).*
org(ab, _, _, _).

A **strong** fault model specifies all the possible ways in which a component can fail [Struss and Dressler, 1989]. In general, a component can have several failure states. An abnormal OR gate, for example, might have the output stuck-at-1 (*s1*) or stuck-at-0 (*s0*):

org(s1, 0, 0, 1).
org(s0, 0, 1, 0).
org(s0, 1, 0, 0).
org(s0, 1, 1, 0).

The **exoneration** principle [Raiman, 1989] is a special case of the strong fault model. It specifies as abnormal any behavior different from normal:

org(ab, X, Y, Z) $\leftarrow \neg$ *or(X, Y, Z).*

Next we define the concepts of a diagnosis and a conflict for $\langle SD, COMPS, OBS\rangle$, assuming that an observation, a ground instance of *OBS*, is given. In the following definitions $\forall F$ denotes universal closure of the formula F, i.e., all free variables in the formula F are universally quantified.

Definition. An *ok-instance* of a term is an instance where some variables are replaced by the constant *ok*. A *ground instance* is an instance where all the variables are replaced by constants.

Definition. A *diagnosis D* for $\langle SD, COMPS, OBS\rangle$ is an instance of *COMPS* such that $SD \models \forall m(D, OBS)$.

Intuitively, a diagnosis is an assignment of states to components such that it is 'consistent' with *OBS* and logically follows from *SD*. States of some components may be irrelevant (note the universal closure).

Definition. A *conflict C* for $\langle SD, COMPS, OBS\rangle$ is an *ok-instance* of *COMPS* such that $SD \models \forall \neg m(C, OBS)$.

The definition of a conflict is standard, i.e., a set of components which cannot be simultaneously *ok*. It can be straightforwardly extended to the definition of a minimal conflict.

Example. Suppose *SD* consists of the weak fault model of the adder, and the observation $OBS = \langle 1,0,1,\ 0,0\rangle$ is given. Then $\langle ok,ok,ok,ok,ab\rangle$ and $\langle ab,ab,A1,A2,O1\rangle$ are diagnoses, while $\langle ok,X2,A1,ok,ok\rangle$ is a conflict.

This characterization of a diagnosis subsumes most of the previous ones. In the consistency-based approach [Reiter, 1987], a diagnosis[1] is a set of abnormal ($\neq ok$) compo-

[1]Reiter's definition of a diagnosis actually includes the minimality criterion and corresponds to our definition of a minimal diagnosis (see subsequent definitions).

nents such that *SD* and *OBS* are consistent with all the other components being *ok*. In Sherlock [de Kleer and Williams, 1989] the definition is extended to include a behavioral mode (state) for each component. In both cases a diagnosis is essentially a ground instance of *COMPS*. However, a diagnosis needs not commit a state to each component when the state is 'don't care' [Poole, 1989]. This led to the definition of a partial diagnosis [de Kleer *et al.*, 1990] which corresponds to a non-ground instance of *COMPS* but, on the other hand, does not include states of components.

Apart from being simple, our definitions are also operational since diagnoses and conflicts can effectively be computed by a logic programming system. The search for a logical consequence of *SD* is realized by the search for an answer substitution Θ such that $SD \cup \{\neg m(A\Theta, OBS)\}$ is unsatisfiable, where A is constrained to be an ok-instance of *COMPS*. If such a substitution exists we can conclude $D = A\Theta$ is a diagnosis. If not, and regarding *SD* under the closed world assumption [Lloyd, 1987], we can conclude that $C = A$ is a conflict. Like in consistency-based diagnosis, A can be interpreted as an assumption that some components are not abnormal, i.e., they are *ok*.

Example. Suppose *SD* is the strong fault model of the adder, and $OBS = \langle 1,0,1,0,0 \rangle$. The following query returns four answer substitutions, i.e., diagnoses:

$\leftarrow$ $A = \langle X1,X2,ok,A2,O1 \rangle$, $adder(A, \langle 1,0,1,\ 0,0 \rangle)$.
$A = \langle ok,ok,ok,ok,s0 \rangle$;
$A = \langle ok,ok,ok,s0,ok \rangle$;
$A = \langle s0,s0,ok,ok,ok \rangle$;
$A = \langle s0,s0,ok,s1,s0 \rangle$

The first three diagnoses are minimal while the fourth is subsumed by the first and the third one, but not by the second one. The following two definitions make the notions of subsumption and minimality precise.

Definition. A diagnosis $D' = \langle S'_1, \ldots, S'_n \rangle$ *subsumes* a diagnosis $D = \langle S_1, \ldots, S_n \rangle$ (we write $D' \subseteq D$) iff $\forall i = 1, \ldots, n\ (S'_i = ok) \vee (S'_i = S_i)$.

Definition. A *minimal* diagnosis is a diagnosis which is subsumed by no other diagnosis.

Note that a minimal diagnosis is always ground since any non-ground diagnosis is subsumed by its ok-instance. Further, for a minimal number of abnormal components, there might be several minimal diagnoses since a component might be assigned different abnormal ($\neq ok$) states.

In the next section we present IDA, an algorithm which actually computes minimal diagnoses. Its distinguishing feature in comparison to most consistency-based algorithms is that it computes minimal diagnoses directly from diagnoses, and not from conflicts. Further, the search for the minimal diagnoses is clearly separated from calls to the model. The separation is realized by a function TP [Reiter, 1987] which implements a call to the underlying theorem prover.

3 Computing minimal diagnoses

It is convenient to represent the search space of diagnoses and conflicts as a subset-superset lattice (Figure 2). The top element of the lattice corresponds to a tuple where all components are $\neq ok$, and the bottom element to the tuple where all components are *ok*. A diagnosis is represented by a set of components which are $\neq ok$, and a conflict, i.e., a set of components which cannot simultaneously be *ok*, by the complement of the set. Note that in this representation a smaller conflict corresponds to a larger set.

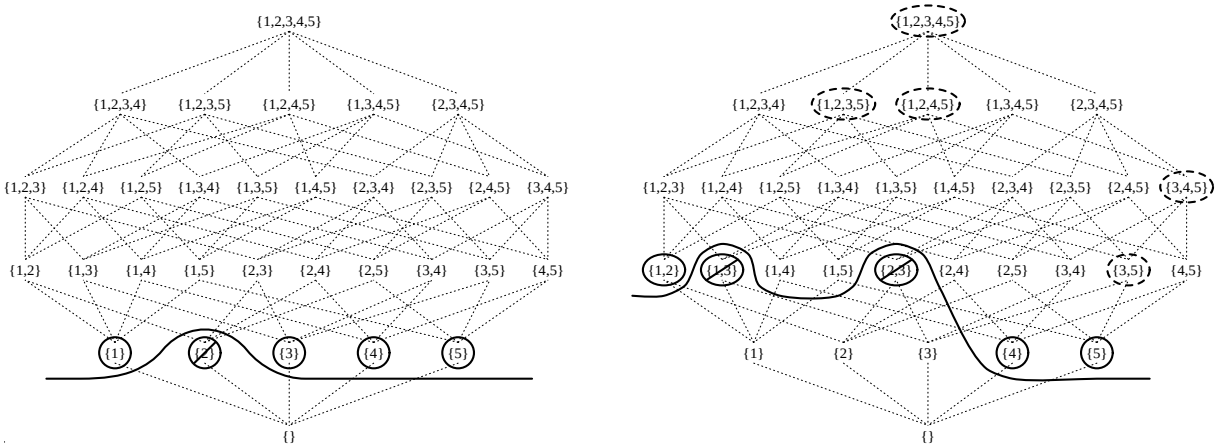

Figure 2: Search lattices for the structural (left), and weak and strong fault model of the adder (right), given the observation $\langle$1,0,1, 0,0$\rangle$. $\oslash$ denote minimal conflicts, and $\bigcirc$ minimal diagnoses. For the structural and weak fault model all supersets of minimal diagnoses are also diagnoses. For the strong fault model only nodes in dashed ovals denote non-minimal diagnoses.

Example. The diagnosis $\langle ab,ab,ok,ok,ok\rangle$ is represented by the set {1,2}, and the conflict $\langle ok,X2,A1,ok,ok\rangle$ by the set {2,3}.

First we define the TP function that takes as arguments *SD*, *OBS*, and a label L, an element of the lattice. TP verifies whether L is a diagnosis or a conflict by calling the model. If the call to the model succeeds, TP returns a diagnosis which is extracted from the answer substitution. If the call fails then L is a conflict. In the following $\neg X$ denotes the complement of a set X, i.e., $\neg X = \{1,\ldots,n\} - X$.

Function TP(*SD*, *OBS*, L)
$A := \langle S_1,\ldots,S_n\rangle \mid S_i = ok,\ i \in \neg L$ (A is an ok-instance of *COMPS*)
if $\exists\Theta$ such that $SD \cup \{\neg m(A\Theta,OBS)\}$ is unsatisfiable
then return $D := \neg\{i \mid S_i = ok,\ S_i \in A\Theta\}$ (D is a diagnosis, $D \subseteq L$)
else return *false* (L is a conflict).

Note that a successful call to TP might return a diagnosis D deep below the label L and that $D \subseteq L$ always holds. This is due to the non-ground calls to the model, i.e., S_j $(j \neq i)$

in A are distinct variables. On the other hand, a label can only be verified whether it is a conflict or not. Our TP function is therefore exactly the opposite of the one defined by Reiter [Reiter, 1987], i.e., the roles of diagnoses and conflicts are reversed.

Reiter's algorithm [Reiter, 1987] searches the lattice bottom-up (from conflicts to diagnoses), in a breadth-first fashion (diagnoses of smaller cardinality are found first). Our algorithm implements a top-down, depth-first search through the lattice. In diagnosing real systems, the search lattice is large and minimal diagnoses are usually near the bottom of the lattice. Depth-first search, coupled with our TP function, allows for deep 'dives' into the lattice and in an average case at least a few diagnoses are found efficiently. Further, by relaxing the problem and by a simple modification to the basic algorithm, we can ensure that the worst case complexity remains polynomial. Before we define the *All_diags* procedure which computes all minimal diagnoses and conflicts, we need one additional definition.

Definition. Suppose Xs is a collection of sets. A *hitting set* H for Xs is a set, such that $H \cap X \neq \{\}$ for each $X \in Xs$. A hitting set is *minimal* iff no proper subset of it is a hitting set.

Function *H(Xs)*
non-deterministically returns a minimal hitting set for a collection of sets Xs.

Procedure *All_diags*(Ds, Cs)
inputs: SD, OBS,
$Ds := \{\}$, $Cs := \{\}$,
outputs: Ds, a set of all minimal diagnoses,
Cs, a set of all minimal conflicts,
while $\exists L\ (L := \neg H(Ds)) \ \wedge \ (L \not\subseteq C,\ C \in Cs)$
do **if** TP(SD, OBS, L) returns D
then *Min_diag*(D, Cs),
$Ds := Ds \cup \{D\}$
else $Cs := Cs \cup \{L\} - \{C \mid C \subset L,\ C \in Cs\}$
(delete non-minimal conflicts C from Cs).

The procedure starts with the label $L = \{1, \ldots, n\}$. In each iteration of the while loop, either a minimal conflict or a minimal diagnosis is found (through a call to the *Min_diag* procedure). A label generated as a complement of a hitting set of previous diagnoses ensures that a new diagnosis will be distinct from the previous ones. The second condition in the while statement ($L \not\subseteq C,\ C \in Cs$) prevents redundant TP calls since any label which is a subset of a conflict is also a conflict. The procedure terminates when all minimal diagnoses and conflicts are found. This is due to the fact that minimal diagnoses are exactly minimal hitting sets of conflicts [Reiter, 1987] and therefore the while loop condition cannot be satisfied any more.

The *Min_diag* procedure is invoked when a diagnosis D, not necessarily minimal, is found. The procedure searches the sub-lattice under D until it finds a minimal diagnosis.

Procedure *Min_diag*(D, Cs)
inputs: *SD*, *OBS*,
D, a diagnosis,
Cs, a set of conflicts,
outputs: D, a minimal diagnosis,
Cs, an updated set of conflicts,
while $\exists L\ (L := D - \{i\},\ i \in D)\ \wedge\ (L \not\subseteq C,\ C \in Cs)$
do **if** TP(*SD*, *OBS*, L) returns D'
then $D := D'$
else $Cs := Cs \cup \{L\} - \{C \mid C \subset L,\ C \in Cs\}$
(delete non-minimal conflicts C from Cs).

At each step the *Min_diag* procedure removes an arbitrary element i from the diagnosis. This corresponds to the assumption that the component i is *ok*. If the TP call succeeds the returned diagnosis is a new candidate for a minimal one. If the TP call fails L is a conflict and i is not removed from D ever again since any subset of a conflict is also a conflict. The procedure terminates either when $D = \{\}$, i.e., all components are *ok*, or when all generated subsets of a diagnosis turn out to be conflicts. The *Min_diag* procedure is similar to the algorithm for computing the first diagnosis by Friedrich *et al.* [Friedrich *et al.*, 1990]. The difference is that we allow for non-ground calls to the model (which enable 'dives' deep down the lattice), and that a minimal diagnosis can be computed from any diagnosis, not just from the top element of the lattice.

Example. Suppose *SD* is the weak fault model of the adder, and *OBS* = ⟨1,0,1, 0,0⟩. The algorithm returns minimal diagnoses Ds = {{4},{5},{1,2}}, and minimal conflicts Cs = {{1,3},{2,3}}. The following is an annotated and abbreviated trace of the algorithm.

call *All_diags*(Ds={}, Cs={})
$L = \neg H(\{\}) = \{1,2,3,4,5\}$
suppose TP(*SD*, *OBS*, {1,2,3,4,5}) returns a diagnosis $D = \{5\}$
(in the worst case the very same label {1,2,3,4,5} could be returned)
call *Min_diag*({5}, {})
$L = \{5\} - \{i{=}5\} = \{\}$
TP(*SD*, *OBS*, {}) returns *false*, {} is a conflict
$Cs = \{\{\}\}$
exit *Min_diag*({5}, {{}})
$Ds = \{\{5\}\}$
$L = \neg H(\{\{5\}\}) = \{1,2,3,4\}$
suppose TP(*SD*, *OBS*, {1,2,3,4}) returns a diagnosis $D = \{4\}$
call *Min_diag*({4}, {{}})

L = {4} − {i=4} = {}, however {} ⊆ C={} and there is no alternative L
exit *Min_diag*({4}, {{}})
Ds = {{4},{5}}
$L = \neg H(\{\{4\},\{5\}\}) = \{1,2,3\}$
suppose TP(*SD*, *OBS*, {1,2,3}) returns a diagnosis D = {1,2,3}
call *Min_diag*({1,2,3}, {{}})
L = {1,2,3} − {i=1} = {2,3}
TP(*SD*, *OBS*, {2,3}) returns *false*, {2,3} is a conflict
Cs = {{2,3}}, note that {} is deleted from *Cs* since {} ⊂ L={2,3}
L = {1,2,3} − {i=2} = {1,3}
TP(*SD*, *OBS*, {1,3}) returns *false*, {1,3} is a conflict
Cs = {{1,3}, {2,3}}
L = {1,2,3} − {i=3} = {1,2}
TP(*SD*, *OBS*, {1,2}) returns a diagnosis D = {1,2}
L = {1,2} − {i=1} = {2}, however {2} ⊆ C={2,3} and thus another L is generated
L = {1,2} − {i=2} = {1}, however {1} ⊆ C={1,3} and there is no alternative L
exit *Min_diag*({1,2}, {{1,3},{2,3}})
Ds = {{4},{5},{1,2}}
$L = \neg H(\{\{4\},\{5\},\{1,2\}\}) = \{1,3\}$, however {1,3} is a conflict and another L is generated
$L = \neg H(\{\{4\},\{5\},\{1,2\}\}) = \{2,3\}$, however {2,3} is also a conflict and there is no other L
exit *All_diags*(*Ds*={{4},{5},{1,2}}, *Cs*={{1,3},{2,3}}

4 Controlling the complexity

If the number of the system components is n there might be $O(2^n)$ minimal diagnoses. In general, attempting to compute *all* minimal diagnosis is asking for more information than one could ever hope to use. [Friedrich *et al.*, 1990] show that even the *next diagnosis* problem is intractable: given a set of already found minimal diagnoses *Ds*, deciding whether a next minimal diagnosis $D \notin Ds$ exists is NP-complete. The statement holds for a weak fault model, and with a strong fault model things get worse since then even deciding whether an *arbitrary diagnosis* exists is NP-complete. This has been shown in [Friedrich *et al.*, 1990] for consistency-based diagnosis, and in [Bylander *et al.*, 1989] for abductive diagnosis.

From the above results we can identify two sources of potential intractability: the search through the lattice, and the type of the fault model used. A nice feature of our algorithm is that it separates the lattice search from the TP calls and therefore enables to address each issue individually.

4.1 Computing the first k diagnoses

Let us first assume that, ignoring the type of the model, a call to TP requires constant time. Then the single source of exponential complexity in our algorithm is the computation of a *minimal hitting set* in the while loop of the *All_diags* procedure. Given a collection Ds of subsets of $\{1, \ldots, n\}$ (already found minimal diagnoses) computing a minimal hitting set $H(Ds)$ (a complement of the label L) is NP-complete [Garey and Johnson, 1979, p. 222]. However, if $|\,Ds\,| = k$ then there is at most n^k hitting sets, and computing a minimal hitting set is in $O(n^k)$. Each label L in the *All_diags* procedure is either a (non-minimal) diagnosis or a minimal conflict. For k diagnoses there is at most n^k minimal conflicts, and computing a label L which is not a conflict requires at most $n^k \times n^k$ comparisons, i.e., is in $O(n^{2k})$. If L is a conflict, the number of diagnoses k does not increase, and computing the next label (or deciding that there is none) remains polynomial. If L is a diagnosis then the *Min_diag* procedure finds a minimal diagnosis in no more then n steps, and produces no more then n conflicts. Therefore, for a fixed set of k diagnoses Ds, the algorithm decides whether the next diagnosis exists (and finds one) in polynomial time $O(n^{2k})$. Note that this does not contradict the result reported in [Friedrich *et al.*, 1990] since no bound on the number of already found diagnoses was set there. However, due to their pessimistic result, a diagnostic algorithm was proposed which computes just the first diagnosis in polynomial time. Our *All_diags* procedure can be trivially modified to the *K_diags* procedure which computes the first k diagnoses in polynomial time.

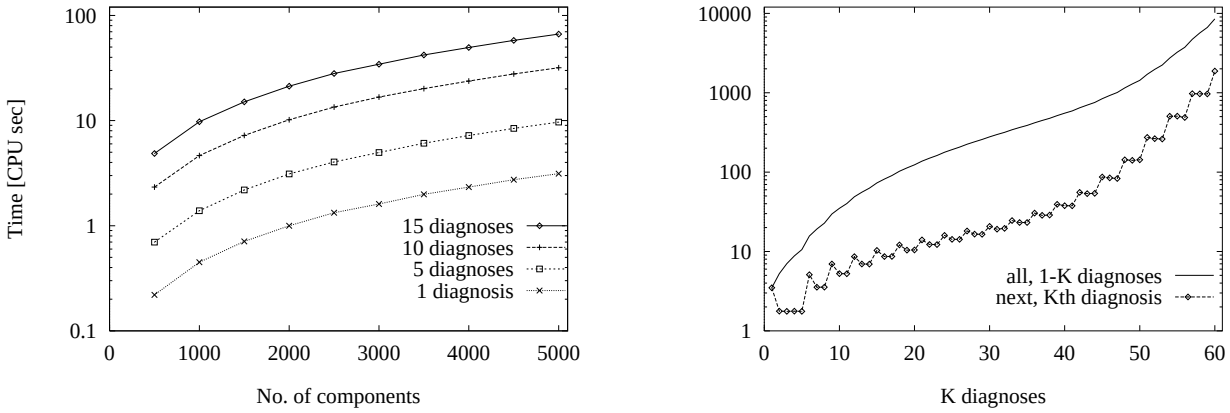

Figure 3: Diagnostic time for the first k diagnoses as a function of the number of components n (left) and as a function of k ($n = 5000$, right) for the m-bit ripple carry adder ($n = 5 \times m$). Logarithmic time scale indicates a sub-exponential increase of time with the number of gates.

In an experimental evaluation, we constructed a model consisting of thousands of components, and measured the CPU time required to find the first k diagnoses (Figure 3). The model is an m-bit ripple carry adder [de Kleer, 1991] consisting of m strong models of the binary adder (see Figure 1). The output $Out2_{i-1}$ of the adder $i-1$ is connected to the

input $In3_i$ of the adder i ($i = 2, \ldots, m$). All inputs and outputs were set to 0, except for the $Out1_m$ of the last adder m which was set to 1 (this is the only faulty output). Note that the inputs are propagated through almost every gate of this circuit, and that an m-bit adder consists of $5 \times m$ gates. Results in Figure 3 indicate that even such large devices can be diagnosed efficiently. De Kleer reported [de Kleer, 1991] that the unfocused GDE [de Kleer and Williams, 1987] can diagnose a 4-bit adder in 60 CPU seconds, while the unfocused Sherlock [de Kleer and Williams, 1989] can do a 6-bit adder in 60 CPU seconds. His new algorithm with focusing can find 5 probable diagnoses for a 500-bit adder (2500 gates) in 6 CPU seconds on Symbolics XL1200 [de Kleer, 1991]. Without focusing, our algorithm IDA computes the first 5 diagnoses (which happen to be the probable ones) in 4 CPU seconds. IDA is implemented in SICStus Prolog [Carlsson and Widen, 1991] and the experiments were run on a SUN IPX workstation[2]. The m-bit ripple carry adder model was specified by a pure Prolog program — no CLP($\mathcal{B}$) or combination of different models was used.

4.2 Combining structural, weak, and strong fault models

Another potential source of combinatorial explosion in diagnosis is the type of the model used, and in this subsection we analyze its impact on the complexity of the TP function. In order to control the overall complexity and to ensure a smooth degradation of performance we propose to use several, increasingly more detailed and computationally demanding models. This is related to the notion of *abstraction* which has been formally defined in the context of theorem proving [Giunchiglia and Walsh, 1989], and to some extend exploited in model-based diagnosis [Gallanti *et al.*, 1989, Mozetič, 1991, Struss, 1992]. A crucial observation is that a class of abstractions, called *truthful*, when applied to a model $M{=}\langle SD, COMPS, OBS\rangle$ yields an abstract model M' where diagnoses are preserved, i.e., $Ds \subseteq Ds'$ [Mozetič, 1991]. An obvious consequence of this is that conflicts are preserved when moving from the abstract M' to the detailed model M, i.e., $Cs' \subseteq Cs$. Here we propose to use three types of models of increasing complexity: a structural model (M''), a weak (M'), and a strong fault model (M).

Given a **structural model**, for any faulty output, a conflict is precisely the set of all components connected (directly or indirectly) to it [Bakker *et al.*, 1989]. For f faulty outputs there is exactly f minimal conflicts, and at most n^f minimal diagnoses. No search is needed to find all conflicts Cs'', and therefore *all* minimal diagnoses Ds'' can be computed in $O(n^f)$ time.

Diagnoses Ds'' are then verified by the weak fault model. Those that are rejected are removed from Ds'' and added to the set of conflicts Cs''. Note that the remaining set Ds'' contains *all* minimal diagnoses of cardinality $\leq f$ for the weak fault model. Additional diagnoses can be computed by calling the *K_diags* procedure with the parameters set to Ds'' and Cs'' instead of empty sets.

[2] IDA is available via anonymous ftp from ftp.ai.univie.ac.at.

A **weak fault model** defines connectivity and normal behavior of components. In order to ensure that no model call requires exponential time, the TP function has to be modified so that all model calls are *ground* [Friedrich *et al.*, 1990]. This is easily achieved by introducing another distinguished constant, say *ab* (which stands for a state $\neq ok$), and instantiating all free variables to *ab*. Nothing else in the algorithm changes, and the complexity analysis remains valid. As a consequence, in addition to all k minimal diagnoses Ds' of cardinality $|\, D' \,| \leq f$, the *K_diags* procedure can compute an additional $k+1$-st minimal diagnosis in $O(n^{2k})$ time.

The minimal diagnoses Ds' can then be verified by the **strong fault model** M. A combination of different abnormal behaviors defined by the strong model is a potential source of combinatorial explosion. For a minimal diagnosis D' there might be $O(2^{|D'|})$ different minimal diagnoses Ds. However, the cardinality of a minimal diagnosis D' is typically small, $|\, D' \,| \ll n$, and therefore the verification is usually tractable. In the worst case it can happen that no minimal diagnosis from Ds' is admitted by the strong fault model. But even then we know that any minimal diagnosis D is of cardinality $|\, D \,| > f$, and we can use the weak fault model to compute the next, $k+1$-st minimal diagnosis.

5 Using CLP($\mathcal{B}$) as a theorem prover

In this section we show how a richer computational domain than the Herbrand universe interacts with the models and the diagnostic algorithm. Whereas we will employ a more powerful theorem prover TP, we will stick with the diagnostic algorithm, in particular with its data structures for the *ground* representation of minimal diagnoses and conflicts.

As far as TP is concerned, we stay in the context of logic programming but extend syntactic unification by solving equations over interpreted terms [Jaffar and Lassez, 1987] — boolean expressions in this particular case. The resulting constraint logic programming language CLP($\mathcal{B}$) was realized in a general framework [Holzbaur, 1990], where the implementation reduces to the *Prolog* formulation of a specialized unification algorithm. There exist several boolean unification algorithms [Crone-Rawe, 1989]. We chose one that was published by [Büttner and Simonis, 1987]; the origin of the method goes back to [Boole, 1947]. The algorithm computes the *most general boolean ring unifier* θ of two terms t_1 and t_2. It operates on a deterministic disjunctive minimal normal form [Martin and Nipkov, 1986] for terms in the boolean ring $\langle V, \oplus, \wedge, 0, 1 \rangle$, where V is the set of variables and $\oplus$ is the *xor* operator.

5.1 Reformulation of the models

Instead of using extensional descriptions of the structural, weak and strong fault models we can formulate them in terms of boolean algebra expressions. For a structural model this is always possible, for a weak fault model the restriction to boolean component descriptions

applies, and for a strong fault model we have the additional restriction that only the exoneration model can be expressed in boolean terms. In the following we encode correct inputs and outputs and the normal state as 0 (zero) instead of *ok*, and the abnormal state and faulty inputs and outputs as 1. For brevity we only show the OR gate reformulations.

- **structural model:** $org(\ (\neg X \wedge \neg Y \wedge Z) \vee W,\ X,\ Y,\ Z)$.
  If the inputs X, Y to the component are correct and the output Z is faulty we conclude that the device is faulty. Otherwise we cannot conclude anything, which is encoded through an unbound boolean variable W.

- **weak fault model:** $org(\ ((X \vee Y) \oplus Z) \vee W,\ X,\ Y,\ Z)$.
  If the boolean function the component computes disagrees with the output, we conclude that the device is faulty. Otherwise we cannot conclude anything.

- **exoneration model:** $org(\ (X \vee Y) \oplus Z,\ X,\ Y,\ Z)$.
  If the boolean function the component computes disagrees with the output, we conclude that the device is faulty. Otherwise we conclude it functions correctly.

The boolean unification algorithm operates on normal form expressions; all boolean functions are expressed in terms of $\oplus$ and $\wedge$.

Example:
$\neg X \rightarrow 1 \oplus X$
$X \vee Y \rightarrow X \oplus Y \oplus X \wedge Y$

Given the observation $\langle 1,0,1,\ 0,0\rangle$, the weak fault model of the adder returns the following answer substitutions which constitute a solved system of boolean equations in the normal form:

$\leftarrow\ adder(\ \langle X1,X2,A1,A2,O1\rangle,\ \langle 1,0,1,\ 0,0\rangle)$.
$X1 = 1 \oplus A \oplus A \wedge W1,$
$X2 = 1 \oplus A \oplus A \wedge W2,$
$A1 = B \oplus B \wedge W3 \oplus W3,$
$A2 = A \oplus A \wedge W4 \oplus C \oplus C \wedge W4 \oplus W4,$
$O1 = B \oplus B \wedge C \oplus B \wedge C \wedge W5 \oplus B \wedge W5 \oplus C \oplus C \wedge W5 \oplus W5$

Note that this answer substitution captures *all* diagnoses, i.e., the whole lattice above the boundary between the diagnoses and conflicts.

5.2 Impact on the diagnostic algorithm

If we employ the diagnostic algorithm unchanged, it will of course compute the same set of minimal diagnoses and conflicts. We can, however, take advantage of the fact that TP, utilizing CLP($\mathcal{B}$) to decide the unsatisfiability of $SD \cup \{\neg m(A\Theta,\ OBS)\}$, computes the *most general* answer substitution Θ. For our algorithm, this means that each call to TP returns either a conflict or a *minimal* diagnosis.

The first simplification of the algorithm consists in dropping the part of the algorithm which refines diagnoses (*Min_diag*).

Example. CLP($\mathcal{B}$) returns the above answer substitutions for the weak fault model of the adder for the first TP(*SD*, *OBS*, {1,2,3,4,5}) call. According to the definition of TP, this is turned into a diagnosis by setting a maximum number of state variables to *ok* (0). In the implemented algorithm this is done sequentially after the model returned the most general answer substitution. Assume that we already succeeded setting *X1, X2, A1* to 0, which is logically equivalent to the model call:

$\leftarrow$ *adder(* $\langle$*X1,X2,A1,A2,O1*$\rangle$*,* $\langle$*1,0,1, 0,0*$\rangle$*), X1=0, X2=0, A1=0.*
$A2 = 1 \oplus C \oplus C \wedge W4,$
$O1 = C \oplus C \wedge W5 \oplus W5$

Next, $A2$ is set to 0, forcing C to 1 and $O1$ to 1, thus yielding the first minimal singleton diagnosis {5}. Similarly, from the label {1,2,3,4} we discover the second minimal diagnosis {4}. In both cases we know that the diagnoses are minimal — there is no need to check the label {}.

Besides not having to explicitly verify the minimality of the diagnoses, there is an additional possibility to take advantage of the generality of the answer substitutions.

Example. After the discovery of the two single faults {4} and {5}, the label {1,2,3} leads to the following call to the model with the answer substitutions:

$\leftarrow$ *adder(* $\langle$*X1,X2,A1,A2,O1*$\rangle$*,* $\langle$*1,0,1, 0,0*$\rangle$*), A2=0, O1=0.*
$X1 = 1,$
$X2 = 1,$
$A1 = W3$

This corresponds to the minimal diagnosis {1,2} — no need to check {1} or {2}. In addition, under the assumption that both $A2$ and $O1$ are *ok* (sublattice {1,2,3}), we known that neither $X1$ nor $X2$ can possibly be *ok*. The answer substitution tells us that they are definitely abnormal (1). In the original algorithm this information is not available and therefore not considered. Through the second modification of the algorithm we do not only compute a minimal diagnosis from a successful call to the model, but also one conflict per definite abnormal state variable. In our example the two conflicts are {1,3} and {2,3}. The two labels would have been tested by the original version of the algorithm.

Therefore, from three calls to the model we got the three minimal diagnoses and even the two conflicts, which in total covers the lattice.

6 Conclusion

We defined a diagnostic algorithm which clearly separates the search through the space of potential diagnoses from the models and their computational domains. The decom-

position enables an independent analysis and control of the computational complexity. IDA presents an efficient and solid skeleton for more sophisticated algorithms. The nondeterministic choice of label L in *All_diags* and *Min_diag*, for example, calls for the incorporation of probabilities of faults to guide the search (like de Kleer's focusing [de Kleer, 1991]). From the probabilities of components' failures one can assign probabilities to alternative labels, and then select the highest probability label first. The incrementality makes it suitable to interleave the diagnostic process with repair [Friedrich *et al.*, 1990, Friedrich *et al.*, 1992], or add different probing strategies [de Kleer and Williams, 1987, de Kleer and Williams, 1989]. The replaceability of the model interpreter allows for the applicability to a broad spectrum of domains — each of them can be modeled either by a widely available Prolog or by an instance of the Constraint Logic Programming (CLP) scheme.

In CLP, syntactic unification is replaced by a more general constraint satisfaction over specific domains. As an example, we illustrated the applicability of CLP($\mathcal{B}$) for compact modelling and diagnosis of combinatorial circuits. Another model interpreter, CLP($\Re$), can solve systems of simultaneous linear (in)equations over $\Re$eals. This is beyond the capabilities of local constraint propagation methods used by ATMS-based systems. CLP($\Re$) was applied to diagnose analog circuits operating under the AC conditions [Mozetič *et al.*, 1991, Novak *et al.*, 1993]. It enables modeling and diagnosis of soft faults — drifts from the nominal parameter values, and computation with parameter tolerances. In a medical application, the heart model in KARDIO [Bratko *et al.*, 1989] was specified by a pure logic program, and recently reformulated in terms of constraints over finite domains [Mozetič and Pfahringer, 1992]. IDA works with any of the above models.

Acknowledgements

This work was supported in part by the Austrian Federal Ministry of Science and Research and in part by the Austrian "Fonds zur Förderung der wissenschaftlichen Forschung" under grant P9426-PHY. Thanks to Bernhard Pfahringer, Carl Uhrik, and Gerhard Widmer for valuable comments, and to Robert Trappl for making this work possible.

References

[Bakker *et al.*, 1989] Bakker, R.R., Hogenhuis, P.A., Mars, N.J.I., van Soest, D.C. Diagnosis of technical systems using design descriptions. *Proc. Second Generation Expert Systems*, pp. 107-116, Avignon, France, 1989.

[Boole, 1947] Boole, G. *The Mathematical Analysis of Logic*, Macmillan, 1947.

[Bratko *et al.*, 1989] Bratko, I., Mozetič, I., Lavrač, N. *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems.* MIT Press, Cambridge, MA, 1989.

[Büttner and Simonis, 1987] Büttner, W., Simonis, H. Embedding Boolean expressions into logic programming. *Journal of Symbolic Computation 4*, pp. 191-205, 1987.

[Bylander *et al.*, 1989] Bylander, T., Allemang, D., Tanner, M.C., Josephson, J.R. Some results concerning the computational complexity of abduction. *Proc. Intl. Conf. on Principles of Knowledge Representation and Reasoning*, pp. 44-54, Toronto, Morgan Kaufmann, 1989.

[Carlsson and Widen, 1991] Carlsson, M., Widen, J. SICStus Prolog User's Manual. Swedish Institute of Computer Science, Kista, Sweden, 1991.

[Cohen, 1990] Cohen, J. Constraint logic programming languages. *Communications of the ACM 33 (7)*, pp. 52-68, 1990.

[Crone-Rawe, 1989] Crone-Rawe, B. Unification algorithms for boolean rings. SEKI Working Paper SWP-89-01, University of Kaiserslautern, Germany, 1989.

[de Kleer, 1991] de Kleer, J. Focusing on probable diagnoses. *Proc. 9th Natl. Conf. on Artificial Intelligence, AAAI-91*, pp. 842-848, Anaheim, CA, MIT Press, 1991.

[de Kleer *et al.*, 1990] de Kleer, J., Mackworth, A.K., Reiter, R. Characterizing diagnoses. *Proc. 8th Natl. Conf. on Artificial Intelligence, AAAI-90*, pp. 324-330, Boston, MIT Press, 1990.

[de Kleer and Williams, 1987] de Kleer, J., Williams, B.C. Diagnosing multiple faults. *Artificial Intelligence 32*, pp. 97-130, 1987.

[de Kleer and Williams, 1989] de Kleer, J., Williams, B.C. Diagnosis with behavioral modes. *Proc. 11th Intl. Joint Conf. on Artificial Intelligence, IJCAI-89*, pp. 1324-1330, Detroit, Morgan Kaufmann, 1989.

[Friedrich *et al.*, 1990] Friedrich, G., Gottlob, G., Nejdl, W. Physical impossibility instead of fault models. *Proc. 8th Natl. Conf. on Artificial Intelligence, AAAI-90*, pp. 331-336, Boston, MIT Press, 1990.

[Friedrich *et al.*, 1992] Friedrich, G., Gottlob, G., Nejdl, W. Formalizing the repair process. *Proc. 10th European Conf. on Artificial Intelligence, ECAI-92*, pp. 709-713, Vienna, John Wiley & Sons, 1992.

[Friedrich and Nejdl, 1990] Friedrich, G., Nejdl, W. MOMO: Model-based diagnosis for everybody. *Proc. 6th IEEE Conf. on AI Applications, CAIA-90*, pp. 206-213, Santa Barbara, CA, IEEE Press, 1990.

[Gallanti *et al.*, 1989] Gallanti, M., Roncato, M., Stefanini, A., Tornielli, G. A diagnostic algorithm based on models at different level of abstraction. *Proc. 11th Intl. Joint Conf. on Artificial Intelligence, IJCAI-89*, pp. 1350-1355, Detroit, Morgan Kaufmann, 1989.

[Garey and Johnson, 1979] Garey, M.R., Johnson, D.S. (1979). *Computers and Intractability.* W.H. Freeman and Co., New York.

[Genesereth, 1984] Genesereth, M.R. The use of design descriptions in automated diagnosis. *Artificial Intelligence 24*, pp. 411-436, 1984.

[Giunchiglia and Walsh, 1989] Giunchiglia, F., Walsh, T. Abstract theorem proving. *Proc. 11th Intl. Joint Conf. on Artificial Intelligence, IJCAI-89*, pp. 372-377, Detroit, MI, Morgan Kaufmann, 1989.

[Holzbaur, 1990] Holzbaur, C. Specification of constraint based inference mechanisms through extended unification. Ph.D. thesis, Dept. of Medical Cybernetics and AI, University of Vienna, Austria, 1990.

[Jaffar and Lassez, 1987] Jaffar, J., Lassez, J.-L. Constraint logic programming. *Proc. 14th ACM Symp. on Principles of Programming Languages*, pp. 111–119, Munich, 1987.

[Lloyd, 1987] Lloyd, J.W. *Foundations of Logic Programming* (Second edition). Springer-Verlag, 1987.

[Martin and Nipkov, 1986] Martin, U., Nipkov, T. Unification in boolean rings. *Proc. 8th Intl. Conference on Automated Deduction*, pp. 506-513, 1986.

[Mozetič, 1991] Mozetič, I. Hierarchical model-based diagnosis. *Intl. Journal of Man-Machine Studies 35 (3)*, pp. 329-362, 1991. Also in Hamscher, W.C., Console, L., de Kleer, J. (Eds.) *Readings in Model-based Diagnosis*, pp. 354-372, Morgan Kaufmann, San Mateo, CA, 1992.

[Mozetič *et al.*, 1991] Mozetič, I., Holzbaur, C., Novak, F., Santo-Zarnik, M. Model-based analogue circuit diagnosis with CLP($\Re$). *Proc. 4th Intl. GI Congress*, pp. 343-353, Munich, Springer-Verlag, 1991.

[Mozetič and Pfahringer, 1992] Mozetič, I., Pfahringer, B. Improving diagnostic efficiency in KARDIO: abstractions, constraint propagation, and model compilation. In E. Keravnou (Ed.), *Deep Models for Medical Knowledge Engineering*, Elsevier, Amsterdam, 1992.

[Novak *et al.*, 1993] Novak, F., Biasizzio, A., Santo-Zarnik, M., Mozetič, I. On automatic fault isolation using DFT methodology for active analog filters. *Proc. European Test Conf., ETC-93*, IEEE Press, 1993.

[Poole, 1989] Poole, D. Normality and faults in logic-based diagnosis. *Proc. 11th Intl. Joint Conf. on Artificial Intelligence, IJCAI-89*, pp. 1304-1310, Detroit, Morgan Kaufmann, 1989.

[Raiman, 1989] Raiman, O. Diagnosis as a trial: the alibi principle. Report, IBM Scientific Center, Paris, 1989.

[Reiter, 1987] Reiter, R. A theory of diagnosis from first principles. *Artificial Intelligence 32*, pp. 57-95, 1987.

[Saraswat *et al.*, 1990] Saraswat, V.A., de Kleer, J., Raiman, O. Contributions to a theory of diagnosis. *Proc. First Intl. Workshop on Principles of Diagnosis*, pp. 33-38, Stanford University, Palo Alto, 1990.

[Struss, 1992] Struss, P. What's in SD? Towards a theory of modeling for diagnosis. In Hamscher, W.C., Console, L., de Kleer, J. (Eds.) *Readings in Model-based Diagnosis*, pp. 419-449, Morgan Kaufmann, San Mateo, CA, 1992.

[Struss and Dressler, 1989] Struss, P., Dressler, O. "Physical negation" — integrating fault models into the general diagnostic engine. *Proc. 11th Intl. Joint Conf. on Artificial Intelligence, IJCAI-89*, pp. 1318-1323, Detroit, Morgan Kaufmann, 1989.