Polynomial-Time Model-Based Diagnosis with the Critical Set Algorithm

Rita L. Childress and Marco Valtorta*
Department of Computer Science
University of South Carolina
Columbia, SC 29208, U.S.A.
{childre,mgv}@usceast.cs.scarolina.edu

Abstract

Mozetic has recently given an algorithm (called IDA) for polynomial-time diagnosis of systems described using models written in Prolog, under some conditions of which the most important is ignorance of abnormal behavior. Mozetic's algorithm uses models of the system to be written in such a way that they can be called with a partial instantiation of their parameters and return a full instantiation, a byproduct of the use of Prolog to write the models. We show that the requirement is not necessary to insure polynomial-time diagnosis. We also show that, by exploiting Loveland's critical set algorithms, we can obtain performance comparable to IDA without requiring IDA-style models.

1 Introduction

There has been much work in recent years on the complexity of consistency-based and abductive model-based diagnosis. No exhaustive review of the literature will be attempted here, but see [1, 4, 5, 12] for some of the main results. A recent paper by Mozetic [10] builds on work by Friedrich et al. [5] and shows that it is possible to compute the first k diagnosis (for fixed k) in polynomial time, when only weak fault models (and possibly physical negation axioms) are used. (For readers familiar with [3], this roughly corresponds to models for which the ignorance of abnormal behavior assumption holds.) Mozetic's solution uses a theorem prover that returns a (non-necessarily minimal) diagnosis through the answer substitution mechanism. We show that the problem solved by Mozetic can also be solved by using Loveland's critical set algorithm [8], without using the answer substitution mechanism and with comparable efficiency. We describe our algorithm and its implementation and give some experimental results comparing it to Mozetic's algorithm.

2 Using Critical Sets in Diagnosis

Consider the problem of consistency-based diagnosis as formulated by Reiter [11]. In this formulation, a diagnosis is a set of abnormal components. In certain situations, every superset of a diagnosis is also a diagnosis (although not minimal)[1]. De Kleer et al [3] call this property the "Minimal Diagnostic Hypothesis," give a (non-syntactic) characterization of the system descriptions for which it holds and some (syntactic) sufficient conditions. One of them is "Ignorance of Abnormal Behavior," which we can describe intuitively as follows (cf. [5]): the Minimal Diagnostic Hypothesis holds when the system description (SD) is a set of formulas of the form

$$\ldots \wedge \neg ab(c_1) \wedge \ldots \wedge \neg ab(c_n) \Rightarrow \ldots \tag{1}$$

*We gratefully acknowledge support from CISE SpA through project "A Study of the Complexity of Abstraction in Diagnosis." Send correspondence pertaining to this paper to Marco Valtorta, mgv@usceast.cs.scarolina.edu (internet), (1)(803)777-3767 (fax), (1)(803)777-4641 (phone).

[1] In contrast to [11] we drop the requirement of minimality for diagnoses, for expository convenience.

where $c_1 \ldots c_n \in COMPONENTS$. Note that $ab(c_i)$ always appears in its negated form on the left-hand side of implications, and can therefore be substituted by a non-negated predicate name, such as $ok(c_i)$, so that (1) is essentially a Horn clause. Usually, $OBS \cup SD$ will then also be a set of Horn clauses.

Let a *fault indicator function* for a given $SD \cup OBS$ be a function that maps $C \subset COMPONENTS$ to 1 if C is a diagnosis for $SD \cup OBS$ and to 0 otherwise.

Note that the value of the fault indicator function, f, for a set of components Δ can be computed by testing if

$$SD \cup OBS \cup \{ab(c_i) \mid c_i \in \Delta\} \cup \{\neg ab(c_i) \mid c_i \in COMPONENTS - \Delta\}$$

is consistent, or, more simply, by Proposition 3.4 in [11],

$$SD \cup OBS \cup \{\neg ab(c_i) \mid c_i \in COMPONENTS - \Delta\} \tag{2}$$

is consistent. If it is, then $f(\Delta) = 1$. If it is not, then $f(\Delta) = 0$.

Following Loveland [8], we define monotonicity and criticality.
A function from the power set of a set U to $\{0, 1\}$ is *binary monotone* if $f(S) = 1 \;\Rightarrow\; f(S_1) = 1\ , \forall S_1 \supseteq S$.

A *critical set* S of a binary monotone function f is a set S such that $f(S) = 1$ and $f(S_2) = 0\ \forall S_2 \subset S \wedge S \neq S_2$.

Loveland [8] calls the problem of finding *a* critical set of a binary monotone function the *critical set problem*. He gives four different algorithms to find one of the critical sets of a monotone function. These polynomial-time algorithms take at most $2r\lceil \log n \rceil$ calls to the binary monotone function f, where r is the number of elements in the critical set and n is the number of elements in the universe U. There are several applications for the critical set problem in Knowledge-Based Systems work. Loveland [8] mentions some problems in knowledge base verification (cf. [7]) and refinement. For our purpose, we note that we can use a critical set algorithm to find a diagnosis for a diagnostic problem with a monotone fault indicator function. Recall that, while satisfiability is NP-hard in the general propositional case and checking consistency is undecidable in general, it is easy to decide whether a set of simple Horn clauses (i.e., with bounded term depth of all arguments and bounded arity) is satisfiable or consistent [5, p.331]. From these facts, it is easy to show that, if $SD \cup OBS$ is a set of simple Horn clauses, then the fault indicator function for $SD \cup OBS$ is binary monotone, the critical sets are minimal diagnoses, and *one* diagnosis for a system described by a set of simple Horn clauses can be found in polynomial time. It is remarkable that neither Bylander et al. [1] nor Friedrich et al. [5] seems to be aware of Loveland's efficient algorithms.

Finding More Critical Sets Suppose that a collection of critical sets has been found. Loveland [8, pp.370–371] gives a combinatorial argument that shows that finding whether there are additional critical sets must take exponential time in the size of the number of elements in U. Friedrich et al. [5, Theorem 2] prove the closely related result that the *next diagnosis* problem is NP-complete. Mozetic and Holzbaur [9] and Mozetic [10] gives a polynomial time algorithm to find the first k diagnoses in polynomial time for a fixed k. Like Bylander et al. and Friedrich et al., they also ignore Loveland's paper.

Fault models may easily destroy the Minimal Diagnostic Hypothesis and the Horn property of a system description. Therefore, even finding *one* diagnosis for a system described by a set of Horn clauses is intractable, when fault models are present. *Physical impossibility axioms*, as defined in [5] do not destroy the Horn property, and they may be a practical alternative to fault models.

3 Description of the Algorithm and Complexity

We have implemented an algorithm in Prolog called IC. We will provide only a high-level description of the algorithm in this section; a large part of the Prolog code implementing the algorithm is given in the Appendix. The algorithm merges Mozetic's IDA [10] and the Critical set algorithm (hence the name IC). There are five critical set algorithms in [8], named algorithms I, II, II′, III, and IV. The first diagnosis computed by IC is computed using critical set algorithm II′ in [8]. A full description of this algorithm can be found in the original reference[2]. The algorithm starts from a set that is a diagnosis, i.e., a set S

[2]Note that Algorithm II in [8] is incorrect. We will present a modified version of algorithm II and proof of its correctness elsewhere.

such that $f(S) = 1$. It divides the set of all components into two roughly equal disjoint sets, then checks whether one of these sets is a diagnosis. If it is, the other half is discarded. If it is not, then the other set must contain elements of the critical set being isolated. Therefore, one element of the critical set is isolated in at most $\log n$ calls to the function f, where n is the number of components in the system to be diagnosed. In the worst case, $r(1 + \log n)$ calls to f are necessary to construct a critical set of size r. This contrasts very favorably with the algorithm used in IDA (and the ones described in [5, 12]), which work by generating all immediate subsets of a diagnosis, check each subset to see whether it is a diagnosis, and repeat this process until a critical set is found. This takes time linear in n. Moreover, in contrast to algorithm II′, the performance of IDA's algorithm decreases with the decreasing size of the critical set being isolated. Small critical sets are likely to be prevalent in many diagnostic applications. (In other words, IC is more efficient when minimal diagnoses are small.)

To find subsequent diagnoses, we apply the same algorithm used in IDA, which Loveland describes (using different terminology) in [8, pp.370-371], although we believe that Mozetic rediscovered the algorithm independently. The basic idea is to find a complement (with respect to the set of all components) of the hitting set of the minimal diagnoses found so far that is also a diagnosis, and then apply the critical set algorithm (again, we chose to implement II′) to this diagnosis to compute a minimal diagnosis (critical set). Loveland notes that the difficult part of this computation is to find a suitable complement of the hitting set of minimal diagnoses. In his words: "the problem of finding a set that contains only new critical sets can be extremely costly even when only one more critical set exists. We emphasize that the complexity of finding a critical set is a function of the number of critical sets already found; the first critical set is not hard to find." Mozetic [10] shows that the first k critical sets (diagnoses) can be found in polynomial time.

A difference between IC and IDA is that the implementation of the fault indicator function (f) in IDA is by using a logic programming model. The model is such that, when queried with a list of components (a *label*) S, it will return the value of $f(S)$ (0 or 1) and, when $f(S) = 1$, it will also return a set of abnormal components $S1$ which may be S itself or a subset of S. (The set $S1$ is found using the answer substitution process.) When a small set $S1$ is found, the computation time for IDA is substantially reduced. However, the model is not required to return a smaller diagnosis than the one it is queried with and no guidelines are given for constructing such a model[3]. This point is illustrated by a detailed example and some experimental results in the following section.

4 Experimental Results

We present results obtained by running IDA and IC with some synthetic models, which consist of a list of numbered components and a list of minimal diagnoses. The main purpose of this section is not to show statistically significant or conclusive experimental results (which are beyond the scope of this paper), but to make the discussion of the relative performance in the previous section of IDA and IC more concrete. In the set of examples that follows a system with 15 components is represented. We present two runs of IDA and one run of IC. The performance of IDA is greatly affected by how close to minimal the diagnosis returned by the model through the answer substitution process is. In one of the IDA runs, the model simply returns the minimal diagnosis directly: possibly most of the components in the label are assigned *ok*. In this case, the model itself solves the critical set problem. This is the best possible case for IDA. In the other IDA run, the model returns the label itself as a diagnosis: all components that are in the label are taken to be abnormal.

We consider a 15-component system with minimal diagnoses $\{1, 2\}$, $\{3, 4\}$, $\{6, 7\}$, $\{5, 6, 9\}$, and $\{3, 7, 11, 13\}$. Here follows an annotated trace of IDA, with the best model definition.

```
3 ?- ida(D,C).
15 : [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] is label
% After being queried with f(S) = f({1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}),
% the model returns S1 = {6,7}, a minimal diagnosis.
1 : [7] is label  %f({7}) = f({6}) = 0: {6,7} is a minimal diagnosis, because
1 : [6] is label  %the fault indicator function for all its subsets has value 0
14 : [1,2,3,4,5,7,8,9,10,11,12,13,14,15] is label
```

[3]Note that if the model returned a minimal diagnosis, it would have to solve the critical set problem.

```
% The label above is a complement of the set of diagnoses found so far, {{6,7}}.
% The model returns {1,2}, a minimal diagnosis.
1 : [2] is label
1 : [1] is label
13 : [2,3,4,5,7,8,9,10,11,12,13,14,15] is label
% The label above is a complement of the set of diagnoses found so far, {{1,2},{6,7}}
1 : [4] is label
1 : [3] is label
12 : [2,4,5,7,8,9,10,11,12,13,14,15] is label
12 : [2,4,5,6,8,9,10,11,12,13,14,15] is label
2 : [6,9] is label
2 : [5,6] is label
11 : [2,4,6,8,9,10,11,12,13,14,15] is label
11 : [2,4,5,6,8,10,11,12,13,14,15] is label
12 : [2,3,5,7,8,9,10,11,12,13,14,15] is label
3 : [3,11,13] is label
3 : [3,7,13] is label
3 : [3,7,11] is label
% The label generated below are complements of the hitting sets of the set of
% minimal diagnoses {{1,2},{3,4},{6,7},{5,6,9},{3,7,11,13}}.  None of them are diagnoses.
11 : [2,3,5,8,9,10,11,12,13,14,15] is label
11 : [2,3,5,7,8,9,10,12,13,14,15] is label
11 : [2,3,5,7,8,9,10,11,12,14,15] is label
11 : [2,3,6,8,9,10,11,12,13,14,15] is label
11 : [2,3,5,6,8,10,11,12,13,14,15] is label
12 : [1,4,5,7,8,9,10,11,12,13,14,15] is label
11 : [1,4,6,8,9,10,11,12,13,14,15] is label
11 : [1,4,5,6,8,10,11,12,13,14,15] is label
11 : [1,3,5,8,9,10,11,12,13,14,15] is label
11 : [1,3,5,7,8,9,10,12,13,14,15] is label
11 : [1,3,5,7,8,9,10,11,12,14,15] is label
11 : [1,3,6,8,9,10,11,12,13,14,15] is label
11 : [1,3,5,6,8,10,11,12,13,14,15] is label

C = [12 : [2,4,5,7,8,9,10,11,12,13,14,15],12 :
[1,4,5,7,8,9,10,11,12,13,14,15],1
1 : [2,4,6,8,9,10,11,12,13,14,15],11 : [2,4,5,6,8,10,11,12,13,14,15],
11 : [2,3,6 ,8,9,10,11,12,13,14,15],11 : [2,3,5,8,9,10,11,12,13,14,15],
11 : [2,3,5,7,8,9,10, 12,13,14,15],11 : [2,3,5,7,8,9,10,11,12,14,15],
11 : [2,3,5,6,8,10,11,12,13,14,15 ],11 : [1,4,6,8,9,10,11,12,13,14,15],
11 : [1,4,5,6,8,10,11,12,13,14,15], 11 : [1,3,6,8,9,10,11,12,13,14,15],
11 : [1,3,5,8,9,10,11,12,13,14,15],11 : [1,3,5,7,8,9, 10,12,13,14,15],
11 : [1,3,5,7,8,9,10,11,12,14,15],11 : [1,3,5,6,8,10,11,12,13,14 ,15]]
D = [2 : [1,2],2 : [3,4],2 : [6,7],3 : [5,6,9],4 : [3,7,11,13]]
```

The next trace is the IDA worst case scenario on the same 15-component system:

```
3 ?- ida(D,C).
15 : [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] is label
%The model verifies that S = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} is a diagnosis,
%and returns S1 = S.
14 : [2,3,4,5,6,7,8,9,10,11,12,13,14,15] is label
13 : [3,4,5,6,7,8,9,10,11,12,13,14,15] is label
12 : [4,5,6,7,8,9,10,11,12,13,14,15] is label
11 : [5,6,7,8,9,10,11,12,13,14,15] is label
10 : [6,7,8,9,10,11,12,13,14,15] is label
9 : [7,8,9,10,11,12,13,14,15] is label
9 : [6,8,9,10,11,12,13,14,15] is label
9 : [6,7,9,10,11,12,13,14,15] is label
8 : [6,7,10,11,12,13,14,15] is label
7 : [6,7,11,12,13,14,15] is label
6 : [6,7,12,13,14,15] is label
5 : [6,7,13,14,15] is label
4 : [6,7,14,15] is label
3 : [6,7,15] is label
2 : [6,7] is label
```

14 : [1,2,3,4,5,7,8,9,10,11,12,13,14,15] is label
13 : [2,3,4,5,7,8,9,10,11,12,13,14,15] is label
12 : [3,4,5,7,8,9,10,11,12,13,14,15] is label
11 : [4,5,7,8,9,10,11,12,13,14,15] is label
11 : [3,5,7,8,9,10,11,12,13,14,15] is label
10 : [3,7,8,9,10,11,12,13,14,15] is label
9 : [3,8,9,10,11,12,13,14,15] is label
9 : [3,7,9,10,11,12,13,14,15] is label
8 : [3,7,10,11,12,13,14,15] is label
7 : [3,7,11,12,13,14,15] is label
6 : [3,7,12,13,14,15] is label
6 : [3,7,11,13,14,15] is label
5 : [3,7,11,14,15] is label
5 : [3,7,11,13,15] is label
4 : [3,7,11,13] is label
13 : [1,2,4,5,7,8,9,10,11,12,13,14,15] is label
12 : [2,4,5,7,8,9,10,11,12,13,14,15] is label
12 : [1,4,5,7,8,9,10,11,12,13,14,15] is label
12 : [1,2,5,7,8,9,10,11,12,13,14,15] is label
11 : [1,2,7,8,9,10,11,12,13,14,15] is label
10 : [1,2,8,9,10,11,12,13,14,15] is label
9 : [1,2,9,10,11,12,13,14,15] is label
8 : [1,2,10,11,12,13,14,15] is label
7 : [1,2,11,12,13,14,15] is label
6 : [1,2,12,13,14,15] is label
5 : [1,2,13,14,15] is label
4 : [1,2,14,15] is label
3 : [1,2,15] is label
2 : [1,2] is label
12 : [2,3,4,5,7,8,9,10,12,13,14,15] is label
11 : [3,4,5,7,8,9,10,12,13,14,15] is label
10 : [3,5,7,8,9,10,12,13,14,15] is label
10 : [3,4,7,8,9,10,12,13,14,15] is label
9 : [3,4,8,9,10,12,13,14,15] is label
8 : [3,4,9,10,12,13,14,15] is label
7 : [3,4,10,12,13,14,15] is label
6 : [3,4,12,13,14,15] is label
5 : [3,4,13,14,15] is label
4 : [3,4,14,15] is label
3 : [3,4,15] is label
2 : [3,4] is label
12 : [2,4,5,6,8,9,10,11,12,13,14,15] is label
11 : [4,5,6,8,9,10,11,12,13,14,15] is label
10 : [5,6,8,9,10,11,12,13,14,15] is label
9 : [5,6,9,10,11,12,13,14,15] is label
8 : [5,6,10,11,12,13,14,15] is label
8 : [5,6,9,11,12,13,14,15] is label
7 : [5,6,9,12,13,14,15] is label
6 : [5,6,9,13,14,15] is label
5 : [5,6,9,14,15] is label
4 : [5,6,9,15] is label
3 : [5,6,9] is label
11 : [2,4,6,8,9,10,11,12,13,14,15] is label
11 : [2,4,5,6,8,10,11,12,13,14,15] is label
11 : [2,3,5,8,9,10,11,12,13,14,15] is label
11 : [2,3,5,7,8,9,10,12,13,14,15] is label
11 : [2,3,5,7,8,9,10,11,12,14,15] is label
11 : [2,3,6,8,9,10,11,12,13,14,15] is label
11 : [2,3,5,6,8,10,11,12,13,14,15] is label
11 : [1,4,6,8,9,10,11,12,13,14,15] is label
11 : [1,4,5,6,8,10,11,12,13,14,15] is label
11 : [1,3,5,8,9,10,11,12,13,14,15] is label
11 : [1,3,5,7,8,9,10,12,13,14,15] is label
11 : [1,3,5,7,8,9,10,11,12,14,15] is label
11 : [1,3,6,8,9,10,11,12,13,14,15] is label
11 : [1,3,5,6,8,10,11,12,13,14,15] is label

C = [12 : [2,4,5,7,8,9,10,11,12,13,14,15],12 : [1,4,5,7,8,9,10,11,12,13,14,15],

```
11 : [2,4,6,8,9,10,11,12,13,14,15],11 : [2,4,5,6,8,10,11,12,13,14,15],
11 : [2,3,6,8,9,10,11,12,13,14,15],11 : [2,3,5,8,9,10,11,12,13,14,15],
11 : [2,3,5,7,8,9,10,12,13,14,15],11 : [2,3,5,7,8,9,10,11,12,14,15],
11 : [2,3,5,6,8,10,11,12,13,14,15],11 : [1,4,6,8,9,10,11,12,13,14,15],
11 : [1,4,5,6,8,10,11,12,13,14,15],11 : [1,3,6,8,9,10,11,12,13,14,15],
11 : [1,3,5,8,9,10,11,12,13,14,15],11 : [1,3,5,7,8,9,10,12,13,14,15],
11 : [1,3,5,7,8,9,10,11,12,14,15],11 : [1,3,5,6,8,10,11,12,13,14,15]]
D = [2 : [1,2],2 : [3,4],2 : [6,7],3 : [5,6,9],4 : [3,7,11,13]]

Yes
```

Now we see IC's performance on the same 15-component system:

```
3 ?- ic(15,[[1,2],[3,4],[6,7],[5,6,9],[3,7,11,13]]).
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] is label
[1,2,3,4,5,6,7,8] is label
[1,2,3,4] is label
[1,2] is label
[1] is label
[2] is label
[2,1] is label
[1,3,4,5,6,7,8,9,10,11,12,13,14,15] is label
[1,3,4,5,6,7,8] is label
[1,3,4,5] is label
[1,3] is label
[1,3,4] is label
[4] is label
[4,1] is label
[4,3] is label
[1,3,5,6,7,8,9,10,11,12,13,14,15] is label
[1,3,5,6,7,8,9] is label
[1,3,5,6] is label
[1,3,5,6,7,8] is label
[1,3,5,6,7] is label
[7] is label
[7,1,3] is label
[7,1,3,5] is label
[7,6] is label
[1,3,5,6,8,9,10,11,12,13,14,15] is label
[1,3,5,6,8,9] is label
[1,3,5] is label
[1,3,5,6,8] is label
[9] is label
[9,6,8,1] is label
[9,6,8,1,3] is label
[9,5] is label
[9,5,3,6] is label
[9,5,3] is label
[9,5,6] is label
[1,3,5,6,8,10,11,12,13,14,15] is label
[1,3,6,8,9,10,11,12,13,14,15] is label
[1,3,5,7,8,9,10,11,12,13,14,15] is label
[1,3,5,7,8,9] is label
[1,3,5,7,8,9,10,11,12] is label
[10,11,12,1,3,5,7,8,9,13,14] is label
[10,11,12,1,3,5,7,8,9,13] is label
[13] is label
[13,10,11,12,1,3] is label
[13,10,11,12,1,3,5,7] is label
[13,10,11,12,1,3,5] is label
[13,7] is label
[13,7,5,10,11] is label
[13,7,5,10,11,12,1] is label
[13,7,3] is label
[13,7,3,12,1,5] is label
[13,7,3,12,1,5,10] is label
[13,7,3,11] is label
[1,3,5,7,8,9,10,11,12,14,15] is label
```

System	IDA best	IC	IDA worst
15 components, 3 minimal diagnoses	17 calls to model	32 calls	51 calls
15 components, 5 minimal diagnoses	32 calls to model	67 calls	83 calls
20 components, 3 minimal diagnoses	17 calls to model	32 calls	66 calls
20 components, 5 minimal diagnoses	32 calls to model	68 calls	107 calls
30 components, 5 minimal diagnoses	62 calls to model	110 calls	181 calls

Table 1: Some experimental results comparing the efficiency of IC and IDA

```
[1,3,5,8,9,10,11,12,13,14,15] is label
[1,3,5,7,8,9,10,12,13,14,15] is label
[1,4,5,6,8,10,11,12,13,14,15] is label
[1,4,6,8,9,10,11,12,13,14,15] is label
[1,4,5,7,8,9,10,11,12,13,14,15] is label
[2,3,5,6,8,10,11,12,13,14,15] is label
[2,3,6,8,9,10,11,12,13,14,15] is label
[2,3,5,7,8,9,10,11,12,14,15] is label
[2,3,5,8,9,10,11,12,13,14,15] is label
[2,3,5,7,8,9,10,12,13,14,15] is label
[2,4,5,6,8,10,11,12,13,14,15] is label
[2,4,6,8,9,10,11,12,13,14,15] is label
[2,4,5,7,8,9,10,11,12,13,14,15] is label

Diagnoses are: [2 : [2,1],2 : [4,3],2 : [7,6],3 : [9,5,6],4 : [13,7,3,11]]

Yes
3 ?- Ξ
```

In Table 1 some additional examples are presented.

Throughout this comparison, we have used only calls to the model (or the fault indicator function f) as complexity measure. Note however that the critical set algorithms are described in an imperative language. (Loveland [8] mentions that they were implemented in Pascal.) Since it is not necessary to use the answer substitution feature of the logic programming models, the whole IC algorithm could easily be coded in an imperative language. Such a version of IC is likely to be much faster than a version requiring logic programming-style models. This is a desirable result for reasons of efficiency.

5 Conclusion

The study of the use of the critical set algorithm in diagnosis is not completed in this paper. A natural extension of this work involve the comparison of the critical set algorithm implemented in IC with the other ones described in [8], using not only synthetic examples, but also real-life or realistic ones.

We emphasize, however, that the Minimal Diagnosis Hypothesis (and equivalently monotonicity of the fault indicator function) does not apply in many realistic situations. For example, the use of (strong) fault models usually destroys it. A less known situation in which the Minimal Diagnostic Hypothesis does not hold is the diagnostic problem described by Gallanti et al. in [6] (also cf. [2]). In extreme synthesis, this problem consists of finding a minimal (in terms of non-zero values) solution to a system of linear equalities. We prove that the fault indicator function for this diagnostic problem is not monotonic by giving a counterexample, using the following quantitative model (from [6]):

$$\begin{cases} 6 = 2\Delta p_1 + 3\Delta p_2 - 4\Delta p_3 - 2\Delta p_4 + 5\Delta p_5 \\ 4 = 2\Delta p_1 + 2\Delta p_2 - 6\Delta p_3 + 8\Delta p_5 \\ 0 = \Delta p_1 - 5\Delta p_3 - 5\Delta p_4 \end{cases}$$

Let f be the fault indicator function. In this case, the system has a solution in which $\Delta p_1 = \Delta p_3 = \Delta p_4 = \Delta p_5 = 0$ and $\Delta p_2 \neq 0$, but $\Delta p1 = \Delta p_2 = \Delta p_3 = \Delta p_4 = \Delta p_5 = 0$ is not a solution. Therefore, $\{p_2\}$ is a diagnosis (hence $f(\{p_2\}) = 1$). However, its superset $\{p_2, p_3\}$ is *not* a diagnosis (and therefore $f(\{p_2, p_3\}) = 0$). This directly contradicts the definition of monotonicity for f. This result shows

that we cannot use a critical set algorithm to find diagnoses efficiently in ODS-PI. We note that Mozetic (in [10]) incorrectly includes this as an example of a problem that can be solved by IDA. In particular, IDA finds the following diagnoses: $\{\{p_2\},\{p_5\},\{p_1,p_3\},\{p_1,p_4\},\{p_3,p_4\}\}$. The actual diagnoses are: $\{\{p_2\},\{p_1,p_3\},\{p_1,p_4,p_5\},\{p_3,p_4,p_5\}\}$. Besides the obvious difference between these two sets, not all supersets of $\{p_2\}$ are diagnoses, since this diagnostic problem is non-monotonic[4].

6 Appendix

```
%
%
%                          IC
%          A Diagnostic Procedure that Uses
%      Igor Mozetic's Incremental Diagnostic Algorithm
%                  in Combination with
%         D.W. Loveland's Critical Set Algorithm
%
%        written in SWI-Prolog (University of Amsterdam)
%
%
%            Rita L. Childress and Marco Valtorta
%              University of South Carolina
%                     April 1993



%----------------------------------------------------------------
%  The following part is a Prolog implementation of Loveland's
%  Algorithm II' for isolating a critical set.
%  [Loveland, Donald W., ``Finding Critical Sets.''  _Journal of
%  Algorithms_ 8, 362-371 (1987)].
%
%  Pseudocode for Algorithm II' (quoted from above article)
%
%  {Assumption: a critical set exists}
%  (1) C <-- {}
%      A <-- U
%      R <-- {}
%  (2) Split A into roughly equal disjoint sets A1 and A2
%          such that |A1| <= |A2|
%  (3) If f(C union R union A1) = 1
%      then
%           A <-- A1
%      else  {A2 contains elements of the critical set being
%               isolated}
%          R <-- A1 union R
%          A <-- A2
%  (4) If |A| > 1
%      then
%          return to step 2
%      else  {An element of the critical set is isolated}
%          C <-- A union C
%          A <-- R
%          R <-- {}
%          if A = {}
%          then
%             C is a critical set
%          else
%             if f(C) = 1
%             then
%               C is a critical set
%             else
%               continue (with step 2)
```

[4]The reader familiar with [6] may wonder whether we are using the numerical model or the (abstract) logical model. The latter is the case, but this is irrelevant here, since $\{p_2,p_3\}$ is not a diagnosis in the abstract logical model, while $\{p_2\}$ is.

```
:- dynamic
    min_diag/1.

% makes an assignment if one argument is unbound;
% if both are bound it tests for equality
assign(A,A).

% sets up split routine
split(A,A1,A2) :-
split_aux(A,[],MidA1,[],A2),
reverse(MidA1,A1).

% split routine
% base case if list splits evenly
split_aux([],A1,A1,A2,A2).
% base case if list doesn't split evenly
split_aux([H|[]],OldA1,A1,A2,A2) :-
append([H],OldA1,A1).
% recursive case
split_aux([H|T],OldA1,A1,OldA2,A2) :-
last(T,Last,NewTail),
append([H],OldA1,MidA1),
append(Last,OldA2,MidA2),
split_aux(NewTail,MidA1,A1,MidA2,A2).

% returns the last element in the list (as a list)  and
% the list minus the last element
last(T,[Last],NewTail) :-
reverse(T,T1),
assign(T1,[Last|RevTail]),
reverse(RevTail,NewTail).

% returns the cardinality of a set
cardinality([],Card,Card).
cardinality([_|T],Old,New) :-
New1 is Old + 1,
cardinality(T,New1,New).

% takes the user's minimal diagnoses and asserts each of them as a min_diag
% used to implement the binary monotone fault indicator function for synthetic tests
assert_diags([]).
assert_diags([H|T]) :-
assert(min_diag(H)),
assert_diags(T).

% tests if C is a superset of any minimal diagnosis
% diag(C) is true iff f(C) = 1 in Loveland's paper
diag(C) :-
   write(C), write_ln(' is label'),  % output for trace and counting
                                     % number of calls to model
   min_diag(M),
   test_subset(M,C).

% Succeeds if first argument is a subset of the second argument
test_subset([],_).
test_subset([H|T],C) :-
   member(H,C),
   test_subset(T,C).

% Step 2 of Loveland's algorithm
step2(A,OldC,OldR,C) :-
split(A,A1,A2),
append(OldC,OldR,B1),
append(B1,A1,Big),!,
step3(Big,OldR,MidR,A1,A2,NewA),
step4(OldC,NewA,MidR,C).
```

```
% Step 3 of Loveland's algorithm
step3(Big,OldR,R,A1,A2,A) :-
(diag(Big),
assign(R,OldR),  % R's value didn't change
assign(A,A1);
append(A1,OldR,Temp),
assign(R,Temp),
assign(A,A2)).

% Step 4 of Loveland's algorithm
step4(OldC,MidA,MidR,C) :-
cardinality(MidA,0,Card),
Card > 1,
step2(MidA,OldC,MidR,C).
step4(OldC,MidA,MidR,C) :-
append(OldC,MidA,C),
assign(A,MidR),
assign(A,[]),  % if A is the empty set
        !.
step4(OldC,MidA,MidR,C) :-
append(OldC,MidA,C1),
assign(A,MidR),
assign(R,[]),
(diag(C1),
        assign(C,C1),  % assign result to return parameter
        !; %(redundant?) cut added
step2(A,C1,R,C)).

% setting up critical set algorithm with U as universe of components,
% and C being a critical set being returned
critical_set(U,C) :-
assign(A,U),
assign(OldC,[]),
assign(R,[]),
step2(A,OldC,R,C).

% interface between Mozetic's IDA and the Critical Set Algorithm
% replaces his min_diag with call to critical set algorithm
min_diag(Succs, Diag0, Confs0, Diag, Confs, Obs) :-
Diag0 = _:Y,
critical_set(Y,Diagnosis), !, % essential cut added
length(Diagnosis,L),
Diag = L:Diagnosis,
Confs = Confs0.  %Our min_diag currently does not change Confs.

% top level goal U is the number of components and Diagnoses are
% the minimal diagnoses in the synthetic models
ic(U, Diagnoses) :-
assert_diags(Diagnoses), %sets up the function f (model)
assert(state_size(U)),
ida(D,_), nl, write('Diagnoses are: '), write_ln(D).


% ------------------------------------------------------------------
%% IDA (c) Copyright 1992, Austrian Research Institute for Artificial Intelligence
%%
%% File:   init
%% Author: Igor Mozetic (igor@ai.univie.ac.at)
%% Date:   April 15, 1992
%
%  Top level loading directives.
%  An interface to the model <SD, Comps, Obs> must be defined in terms of
%  the following predicates (see examples):
%
%  model( Comps, Obs )     - a binary predicate which relates Comps to Obs,
%  observation( Obs )      - an instance of Obs (optional),
%  state_functor( Funct )  - a functor of Comps,
```

```
%  state_size( N )         - the arity of Comps,
%  state_normal( I, Ok )   - a constant which denotes normal state in Comps
%                            at argument position I (must be singleton),
%  state_abnormal( I, Ab ) - a constant which denotes abnormal state in Comps
%                            (there can be several abnormal states at I).

:- op( 600, xfy, ':' ).

load_file( File ) :- consult( File ).   % C-Prolog
%load_file( File ) :- compile( File ).   % Quintus, SICStus Prolog

%:- load_file( 'kdiags.pl' ).  % we don't use Mozetic's kdiags.pl
                               % the predicates needed from kdiags.pl are
                               % below in this file, with modifications as noted
:- load_file( 'hset.pl' ).
:- load_file( 'lattice.pl' ).
:- load_file( 'ords.pl' ).   % sets represented by ordered lists
%:- load_file( 'bits.pl' ).   % sets represented by bit vectors

ida( Diags, Confs ) :-
   observation( Obs ),
   k_diags( 1000, Obs, Diags, Confs ),
   retractall(min_diag(_)).
observation([]). % No observations

% sets up to find k = 1000 diagnoses
ida( Obs, Diags, Confs ) :-
   k_diags( 1000, Obs, Diags, Confs ).

k_diags( K, Obs, Diags, Confs ) :-
   k_diags( 0, K, [], [], Diags, Confs, Obs ).

%  k_diags( +N, +K, +Diags0, +Confs0, -Diags, -Confs, +Obs )
%  Given an initial set of minimal Diags0 and (non-minimal) Confs0,
%  computes next K-N minimal diagnoses. Returns updated Diags and Confs.

k_diags( N, K, Diags0, Confs0, Diags, Confs, Obs ) :-
   N < K,
   gen_label( Diags0, Confs0, Label ), !,
   verify_label( Label, Diags0, Confs0, Diags1, Confs1, Obs ),
   ( Diags0 == Diags1 -> N1 = N ; N1 is N+1 ),
   k_diags( N1, K, Diags1, Confs1, Diags, Confs, Obs ).
k_diags( _, _, Diags, Confs, Diags, Confs, _ ).

%  verify_label( +Label, +Diags0, +Confs0, -Diags, -Confs, +Obs )
%  Verifies whether Label is a diagnosis or a conflict. If it is
%  a (non-minimal) Diag0 then min_diag/6 returns a minimal Diag and
%  previous Diags0 are updated into new Diags. Otherwise Label is
%  a Conf, and Confs0 are updated into Confs.

verify_label( Label, Diags0, Confs0, Diags, Confs, Obs ) :-
   call_model( Label, Diag0, Obs ), !,
%   gen_succs( Diag0, Succs ),  % part of IDA not used in our program
   min_diag( Succs, Diag0, Confs0, Diag, Confs, Obs ), % uses our version to
                                                       % call the critical set
                                                       % algorithm
   ord_insert( Diags0, Diag, Diags ).
%   latt_insert( Diags0, Diag, Diags ). % part of IDA not used in our program
verify_label( Conf, Diags, Confs0, Diags, Confs, _ ) :-
   latt_replace( Confs0, Conf, Confs ).

call_model(Label, Label, Obs) :-
Label = _:Y,
diag(Y).

%  gen_label( +Diags, +Confs, -Label )
%  Label is a complement of a hitting set of minimal Diags,
%  such that it is not a SUBset of any Confs.
```

```
%  [2:[1,2], 2:[1,5]] x 4:[1,2,3,4] -> 3:[1,3,4], 4:[2,3,4,5] -> 4:[2,3,4,5]
%  Nondeterministic, NP complexity !

gen_label( Diags, Confs, Label ) :-
   hitting_set( Diags, Hset ),
   ords_complement( Hset, Label ),
   \+ latt_member( Label, Confs ).
```

References

[1] Bylander, Tom, Dean Allemang, Michael C. Tanner, and John R. Josephson. "The Computational Complexity of Abduction." *Artificial Intelligence*, 49, pp.25–60, May 1991.

[2] Childress, Rita, Marco Valtorta, and Giorgio Tornielli. "The Complexity of Diagnosing Continuous Processes with ODS-PI." *Working Notes of the AAAI Workshop on Approximations and Abstractions of Computational Theories*, pp.39–46, San Jose, CA, July 1992. (Also available as Department of Computer Science Report TR9201, University of South Carolina, Columbia, SC, U.S.A.)

[3] de Kleer, Johan, Alan K. Mackworth, and Raymond Reiter. "Characterizing Diagnoses and Systems." In: Walter Hamscher, Luca Console, and Johan de Kleer (eds.). *Readings in Model-Based Diagnosis.* San Mateo, CA: Morgan-Kaufmann, 1992, pp.54–65. (Reprinted from *Artificial Intelligence*, 56 (1992).)

[4] Eiter, Thomas and Georg Gottlob. "The Complexity of Logic-Based Abduction." Christian Doppler Laboratory for Expert System Report CD-TR 92/35, Technical University of Vienna, 1992.

[5] Friedrich, Gerhard, Georg Gottlob, and Wolfgang Nejdl. "Physical Impossibility instead of Fault Models." *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, pp.331–336.

[6] Gallanti, Massimo, Marco Roncato, Alberto Stefanini, and Giorgio Tornielli. "A Diagnostic Algorithm based on Models at Different Levels of Abstraction." *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, pp.1350-1355.

[7] Loveland, Donald W. and Marco Valtorta. "Detecting Ambiguity: An Example in Knowledge Evaluation." *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Karlsruhe, Federal Republic of Germany, pp.182–184.

[8] Loveland, Donald W. "Finding Critical Sets." *Journal of Algorithms*, 8 (1987), pp.362–371.

[9] Mozetic, Igor and Christian Holzbaur. "Controlling the Complexity in Model-Based Diagnosis." Austrian Institute for Artificial Intelligence Report TR-91-3, 1991.

[10] Mozetic, Igor. "A Polynomial-Time Algorithm for Model-Based Diagnosis." Austrian Institute for Artificial Intelligence Report TR-92-26, 1992. (A shortened version of this paper appears in *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-92).*)

[11] Reiter, Raymond. "A Theory of Diagnosis from First Principles." *Artificial Intelligence*, 32 (1987), pp.57–95.

[12] Selman, Bart and Hector J. Levesque. "Abductive and Default Reasoning: A Computational Core." *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, pp.343–348.