Four-Valued Diagnoses for Stratified Knowledge-Bases

Ofer Arieli and Arnon Avron

Department of Computer Science
School of Mathematical Sciences
Tel-Aviv University
Ramat-Aviv 69978, Israel
Email: {ofera,aa}@math.tau.ac.il

Abstract. We present a four-valued approach for recovering consistent data from inconsistent set of assertions. For a common family of knowledge-bases we also provide an efficient algorithm for doing so automaticly. This method is particularly useful for making model-based diagnoses.

1 Introduction

It is well-known that the classical calculus allows only trivial reasoning in the presence of inconsistency. This property is particularly problematic when the system under consideration is aimed to deal with conflicts. This is the case, for instance, with diagnostic systems that are supposed to explain the discrepancy between the actual behavior of some device and the way it is meant to behave. A common approach of handling inconsistent information is to consider some consistent subsets that still contain meaningful data. The usual method of doing so is to consider the maximal consistent subsets of the "polluted" data. The main drawback of this method is that none of these subsets necessarily correspond to the intended semantics of the original information. Even in the simplest inconsistent knowledge-base $KB = \{p, \neg p\}$ every maximal consistent subset of KB classically *contradicts an explicit data of* KB. In the case of diagnostic systems this means that a diagnosis based on a maximal consistent subset might not truthfully determine why a given system is not functioning as it was intended. One might, of course, use the *intersection* of all the maximal consistent subsets. This, however, might be very expensive.

We propose here a different approach to "salvage" consistent data without contradicting any assertion of the original information. Our approach is based on the idea of *reducing* the number of models by using a second order relation (see details below). For a common family of knowledge-bases we also provide an efficient algorithm for recovering this data. We then illustrate the ideas in a diagnostic system for checking faulty circuits. The underlying formalism is based on Belnap's four-valued logic [Be77a,Be77b], and it is nonmonotonic and paraconsistent [dC74] in nature.

2 Preliminaries

We present a formalism that is based on Belnap's well-known four-valued logic. For a detailed discussion on this logic see, e.g., [Be77a,Be77b]. We denote by t and f the classical values. $\bot$ and $\top$ denote, respectively, lack of knowledge and "over"-knowledge (conflict). It is usual to consider these four values according to two partial orders: One, $\leq_t$, might intuitively be understood as reflecting differences in the "measure of truth" that every value represents. According to this order, f is the minimal element, t is the maximal one, and $\bot, \top$ are two intermediate values that are incomparable. $(\{t, f, \top, \bot\}, \leq_t)$ is a distributive lattice with an order reversing involution $\neg$, for which $\neg\top = \top$ and $\neg\bot = \bot$. We shall denote the meet and the join of this lattice by $\wedge$ and $\vee$, respectively. The other partial order, $\leq_k$, is understood (again, intuitively) as reflecting differences in the amount of *knowledge* or *information* that each truth value exhibits. Again, $(\{t, f, \top, \bot\}, \leq_k)$ is a lattice where $\bot$ is its minimal element, $\top$ – the maximal element, and t, f are incomparable.

A double-Hasse diagram with the four elements and the two lattices is given in Figure 1 below.

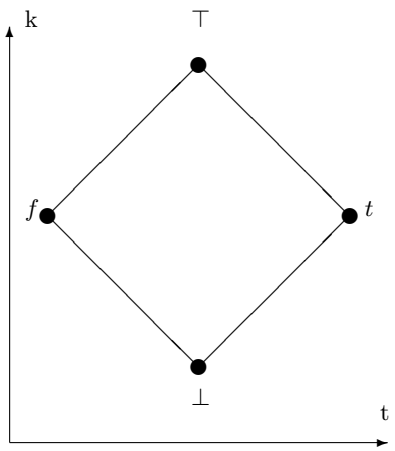

Fig. 1. The four-valued structure

The language we treat here is the propositional language based on $\{\neg, \vee, \wedge, \bot, \top\}$.[1] Given a set S of propositional formulae, we shall denote by $\mathcal{A}(S)$ the set of the atomic formulae that occur in S, and by $\mathcal{L}(S)$ the set of the literals that occur in S. The semantic notions are natural generalizations to the four-valued case of similar classical notions: A *valuation* ν is a function that assigns a

[1] t and f are definable in this language: $f = \top \wedge \bot$ and $t = \top \vee \bot$. $\wedge$ is of course also definable, using de-Morgan law.

truth value from $\{\top, \bot, t, f\}$ to each atomic formula. Any valuation is extended to complex formulae in the standard way. We shall sometimes write $\psi : b \in \nu$ instead of $\nu(\psi) = b$. We will say that ν *satisfies* ψ, iff $\nu(\psi) \in \{t, \top\}$. t and $\top$ are called *designated values*. A valuation that satisfies every formula in a given set of formulas S is said to be a *model* of S. The set of the models of S will be denoted $mod(S)$. Note that unlike in the classical calculus, there are no tautologies here. In fact, excluded middle is not a valid rule in the four-valued case.

The formulae on which we are going to concentrate here are clauses, i.e.: disjuncts of literals. As the following lemma shows, by doing so we do not reduce the generality.

Lemma 1. For every formula ψ there is a finite set S of clauses such that for every valuation ν, $\nu(\psi) \in \{\top, t\}$ iff $\nu(\phi) \in \{\top, t\}$ for every $\phi \in S$.

Proof: By an induction on the structure of ψ. The proof is similar to that of the classical case. Using the fact that de-Morgan's laws, distributivity, commutativity, associativity, and the double negation rule ($\neg\neg\varphi = \varphi$) remain valid in the four-valued case, we can transform any formula into an equivalent one in conjunctive normal form. The lemma follows now from the fact that $\phi_1 \wedge \phi_2$ is designated here iff both ϕ_1 and ϕ_2 are designated. □

Lemma 2. Let ψ be a clause, l_i $(i = 1 \ldots n)$ – its literals, and ν – a valuation on $\mathcal{A}(\psi)$. Then $\nu(\psi) \in \{t, \top\}$ iff there is an $1 \leq i \leq n$ s.t. $\nu(l_i) \in \{t, \top\}$.

Proof: Immediate from the fact that $\{f, \bot\}$ is closed under disjunction. □

Definition 1. A *knowledge-base KB* is a pair $(S, Exact)$, where S is a set of clauses, and $Exact$ is a set of atoms in $\mathcal{A}(S)$ that are assumed to have only classical values. $mod(KB) = mod(S, Exact)$ denotes the set of *exact models* of S, i.e.: the models of S in which every element of $Exact$ is assigned a classical value. Formally: $mod(S, Exact) = \{M \in mod(S) \mid \forall p \in Exact\ M(p) \in \{t, f\}\}$.

We introduced the set Exact because there are cases in which we do not want to leave room to any doubts. For example, what a law says about something should be very clear; It might not be very obviouse, however, if the law is obayed[2] (we shall give a concrete example in Section 5).

Definition 2. Let $M \in mod(S)$. Define: $Inc_M(S) = \{p \in \mathcal{A}(S) \mid M(p) = \top\}$.

Definition 3. Let M, N be two exact models of a knowledge-base $KB = (S, Exact)$.
a) M is *more consistent* than N ($M >_{con} N$) iff $Inc_M(S) \subset Inc_N(S)$. M is *smaller* than N ($M \leq_k N$) iff for any $p \in \mathcal{A}(S)$, $M(p) \leq_k N(p)$.
b) mcem(KB), kmin(KB), and $\Omega(KB)$ respectively denote the set of the *most consistent* exact models of KB (mcems, for short), the set of the k-minimal exact models of KB, and the set of the k-minimal models among the elements of mcem(KB) (*minimal mcems*, for short).

[2] The use of the set Exact is actually a kind of integrity constraint that we force on the system.

Definition 4. Let $KB=(S, Exact)$ be a knowledge-base, and ψ – a formula.
a) $KB \models \psi$ if every exact model of KB is a model of ψ.
b) $KB \models_{mcem} \psi$ if every mcem of KB is a model of ψ.
c) $KB \models_{kmin} \psi$ if every k-minimal exact model of KB is a model of ψ.
d) $KB \models_{\Omega} \psi$ if every minimal mcem of KB is a model of ψ. [3]

Note: The consequence relations $\models_{mcem}$, $\models_{kmin}$, and $\models_{\Omega}$ are preferential logics in the sense of Shoham [Sh87,Sh88]; Such consequence relations are based on the idea that inferences should not take into account every model of a given theory, but only a subset of them, determined according to a certain preference criteria. (Preferential logics has recently received a considerable attention. See, e.g., [KLM90,Pr91,LM92,KL92]).

Example 1. Consider the knowledge-base $KB=(S, Exact)$ where $S=\{p,\ \neg p \vee \neg q\}$, and $Exact=\emptyset$. The single (k-minimal) mcem of KB is $M=\{p:t,\ q:f\}$. M and $N=\{p:\top,\ q:\bot\}$ are the k-minimal models of KB. Thus $KB \models_{mcem} \neg q$ and $KB \models_{\Omega} \neg q$, while $KB \not\models_{kmin} \neg q$ and $KB \not\models \neg q$. When $Exact=\{q\}$, M remains the (minimal) mcem of KB, but now it is also the single element of kmin(KB), therefore $KB \models_{kmin} \neg q$.

Several consequence relations similar to $\models_{mcem}$ are considered in the literature. Priest [Pr91] uses a similar consequence relation $\models_{LPm}$ for defining the logic LPm from the three-valued logic LP. In [AA95] it is shown that $\models_{mcem}$ and $\models_{\Omega}$ are the same in case $Exact=\emptyset$. The proof there applies in the general case as well. Therefore, when switching to four valued semantics and using only the k-minimal mcems, one might consider fewer models than in the case of LPm, since for every k-minimal mcem that assigns $\bot$ to atomic formulae p_i $i=1,...n$, there are 2^n corresponding minimaly consistent models of LP, each one assigns either t or f to these p_i. Moreover, three valued reasoning can be simulated in our framework, since the entailment $KB \models_{LPm} \psi$ is equivalent to $KB, p_1 \vee \neg p_1, \ldots p_n \vee \neg p_n \models_{mcem} \psi$, where $\mathcal{A}(KB)=\{p_1, \ldots, p_n\}$.

Another difference between the present work and [Pr91] is that Priest considers, in fact, only the case $Exact=\emptyset$.

Kifer and Lozinskii [KL92] also consider a similar relation in the framework of annotated logics. Like Priest, they only consider the most consistent models among *all* the possible models. They do not restrict the attention to some relevant subset (as we do) by constraining them in the meta-level. Further discussion and a comparison between $\models_{mcem}$ and the consequence relation of [KL92] can be found in [AA94,AA96].

A basic property of the knowledge-bases that we use here is that for every exact model there is an mcem which is at least as consistent. For finite knowledge-

[3] One can view the consequence relation $\models_{\Omega}$ as a composition of the relations $\models_{mcem}$ and $\models_{kmin}$. First we confine ourselves to the mcems of KB by using $\models_{mcem}$, then we minimize the valuations that we have got by using $\models_{kmin}$.

bases this is trivialy the case. The following proposition assures that this property holds in *every* propositional knowledge-base:

Proposition 1. (Lin's Lemma, [Pr91]) Let KB be a (possibly infinite) set of clauses. For every exact model M of KB there is an mcem M' of KB s.t. $M' \geq_{con} M$. [4]

3 Recovery of knowledge-bases

In this section we describe what we mean by saying "recovering an inconsistent knowledge-base". In particular we define and characterize the recovered parts of a knowledge-base. For that we first have to expand the notion of "consistency" to the four-valued case:

Definition 5. Let S be a set of clauses.
a) A model M of S is *consistent* if $Inc_M(S) = \emptyset$.
b) S is *consistent* if it has a consistent model.
c) $KB = (S, Exact)$ is *consistent* if S has a consistent exact model.

Lemma 3. S is consistent iff it is classically consistent.

Proof: One direction is obvious. For the other, assume that M is a consistent model of S. Then there is no $p \in \mathcal{A}(S)$ s.t. $M(p) = \top$. Consider the valuation M' defined for every $p \in \mathcal{A}(S)$ as follows: $M'(p) = f$ if $M(p) \in \{f, \bot\}$, and $M'(p) = t$ otherwise. By Lemma 2, M' is a model of S as well. □

Definition 6. A subset $S' \subseteq S$ is *consistent in* S w.r.t. $Exact$ if S' is a consistent set that has a consistent exact model M', and there is a (not necessarily consistent) exact model M of S s.t. $M(p) = M'(p)$ for every $p \in \mathcal{A}(S')$.

Example 2. $S' = \{p\}$ is a consistent set, but it is *not* consistent in $S = \{p, \neg p\}$ w.r.t. any set $Exact$, since there is no consistent model of S' that is expandable to a model of S. Similarly, $S' = \{p\}$ is consistent in $S = \{p,\ \neg p \vee q,\ \neg p \vee \neg q\}$ w.r.t. $Exact = \{p\}$, but it is not consistent in S w.r.t. $Exact = \{q\}$, since there is no *consistent* exact model of S' that is expandable to an *exact* model of S.

Definition 7. Let M be an exact model of a knowledge-base $KB = (S, Exact)$. The set that is *associated with* M is: $KB_M = \{\psi \in S \mid \mathcal{A}(\psi) \cap Inc_M(S) = \emptyset\}$.

Example 3. Consider the knowledge-base $KB = (S, \{e\})$ where $S = \{p,\ q,\ \neg p \vee r,\ \neg q \vee \neg r,\ p \vee s,\ \neg r \vee e,\ \neg r \vee \neg e\}$. $M = \{p : \top,\ q : t,\ r : f,\ s : \bot,\ e : t\}$ is an exact model of KB, and $KB_M = \{q,\ \neg q \vee \neg r,\ \neg r \vee e,\ \neg r \vee \neg e\}$.

Proposition 2. A set that is associated with an exact model of KB is consistent in KB.

[4] This lemma is proved in [Pr91] for the three-valued case, and under the implicit assumption that $Exact = \emptyset$. However, it is easy to prove this lemma in our case as well by the same method.

Proof: Let M be an exact model of $KB = (S, Exact)$ and suppose that M' is its reduction to $\mathcal{A}(S) \setminus Inc_M(S)$ only. Obviously, $KB_M \subseteq S$. It is a consistent set in KB, since M' is a consistent exact model of KB_M that is expandable to an exact model (M) of KB. □

Definition 8. A *recovered set* of $(S, Exact)$ is a maximal subset of S that is consistent in S w.r.t. $Exact$.

Example 4. Consider again Example 3. KB_M is a recovered set of KB, since it is a maximal subset of KB that has a consistent model ($\{q:t,\ r:f,\ s:\bot,\ e:t\}$) which is expandable to a model (M) of KB.

Proposition 3. Every recovered set of KB is associated with an mcem of KB.

Proof: Suppose that S' is any set that is consistent in a knowledge-base $KB = (S, Exact)$. Let N' be a consistent exact model of S', and N – its expansion to the whole S. Consider any mcem M that satisfies $N \leq_{con} M$ (by Proposition 1 such a valuation exists). Since $\mathcal{A}(S') \subseteq \mathcal{A}(S) \setminus Inc_N(S) \subseteq \mathcal{A}(S) \setminus Inc_M(S)$, every formula $\psi \in S'$ consists only of literals that are assigned consistent truth values under M. Hence $S' \subseteq KB_M$. Proposition 2 assures that KB_M is consistent in KB, hence $S' = KB_M$ in case S' is maximal. □

Next we provide a condition that implies the existence of a nonempty recovered set for a given knowledge-base:

Proposition 4. Let $KB = (S, Exact)$ be a knowledge base, and suppose that there is an $l \in \mathcal{L}(S)$ s.t. $KB \models_{mcem} l$ and $KB \not\models_{mcem} \bar{l}$. Then there is a nonempty recovered set for KB.

Proof: Without a loss of generality, assume that $KB \models_{mcem} p$ and $KB \not\models_{mcem} \neg p$. Then there is an $M \in mcem(KB)$ s.t. $M(p) \in \{t, \top\}$ while $M(\neg p) \notin \{t, \top\}$, i.e. $M(p) = t$. Consider the set KB_M. It cannot be empty, since otherwise every $\psi \in S$ contains some element of $Inc_M(S)$ or its negation. Define: $N = \{r:f \mid r \in \mathcal{A}(S) \setminus Inc_M(S)\} \cup \{s:\top \mid s \in Inc_M(S)\}$. By Lemma 2, N is an exact model of KB. Moreover, N is an *mcem* of KB, since $Inc_N(S) = Inc_M(S)$. But $N(p) = f$, and so $KB \not\models_{mcem} p$ – a contradiction. Therefore KB_M is a nonempty set, and by Proposition 2 it is consistent in KB. Now, if KB_M is a maximal set with this property then it is the required recovered set of KB, otherwise it is included in a recovered set of KB which cannot be empty. □

4 Stratified knowledge-bases and their recovered sets

In general, computing mcems for a given knowledge-base and discovering its recovered sets might not be an easy task. Even in relatively simple cases, where S is consistent and $Exact = \mathcal{A}(S)$, finding a recovered set for $(S, Exact)$ reduces to the problem of logical satisfaction, since in this case one has to provide a classical model for S. Therefore, we confine ourselves to a special (nevertheless common) family of knowledge-bases, for which we provide an efficient algorithm that computes recovered sets.

Definition 9. Let S be a set of formulae. S_ν — the *dilution* of S w.r.t. a given partial valuation ν — is constructed from S by the following transformations:

1. Deleting every $\psi \in S$ that contains $\top$ or a literal l s.t. $\nu(l) \in \{t, \top\}$,
2. Removing from every formula other than $\bot$ that is left every occurrence of $\bot$ and every occurrence of a literal l such that $\nu(l) \in \{f, \bot\}$. [5] [6]

Proposition 5. If ν can be extended to an exact model of S then S_ν has an exact model. Moreover, the union of ν with any exact model of S_ν is an exact model of S.

Definition 10. Let S be a set of assertions. An atom $p \in \mathcal{A}(S)$ is called a *positive* (*negative*) *fact* of S if $p \in S$ ($\neg p \in S$). p is called *strictly* positive (negative) fact of S if it is a positive (negative) fact of S and $\neg p \notin S$ ($p \notin S$).

Definition 11. A knowledge-base $KB = (S, Exact)$ is called *stratified*, if there is a sequence of "stratifications" $S_0 = S$, S_1, S_2, ..., $S_n = \emptyset$, so that:
a) No S_i $(0 \leq i \leq n)$ contains a fact p s.t. $\{p, \neg p\} \subseteq S_i \cap Exact$.
b) For every $i < n$ there is a (positive or negative) fact $p_i \in \mathcal{A}(S_i)$ s.t. S_{i+1} is the dilution of S_i w.r.t. the partial valuation $p_i : t$ iff p_i is a strictly positive fact, $p_i : f$ iff p_i is a strictly negative fact, and $p_i : \top$ iff p_i is both a positive and a negative fact of S_i.

In all the examples given here, as well as in most of the known puzzles of the literature, the involved knowledge-bases are stratified.

Proposition 6. Let $S_0, S_1, \ldots, S_n$ be a stratification of a knowledge-base KB. For every $0 \leq i \leq n-1$ let ν_i be the partial valuation according to which S_{i+1} is obtained from S_i (I.e., $S_{i+1} = (S_i)_{\nu_i}$). Then $M = \cup_{i=0}^{n-1} \nu_i$ is a model of KB_i.

Proof: By an induction on the structure of a formula in KB. □

Note: Had the dilution of each stratification level been performed w.r.t. more than a single atom (cf. Definition 11b), Proposition 6 wouldn't have been valid anymore. To see this consider, e.g., $KB = (S, \emptyset)$ where $S = \{p,\ q,\ \neg p \vee \neg q\}$. A dilution of S w.r.t. both p and q would have yield a valuation $\nu = \{p : t, q : t\}$, which is not an (exact) model of KB.

Example 5. Let $KB = (S, \{e\})$ be the same knowledge-base of Examples 3 and 4. A possible stratification of S is $S_0 = \{p,\ q,\ \neg p \vee r,\ \neg q \vee \neg r,\ p \vee s,\ \neg r \vee e,\ \neg r \vee \neg e\}$, $S_1 = \{p,\ \neg p \vee r,\ \neg r,\ p \vee s,\ \neg r \vee e,\ \neg r \vee \neg e\}$, $S_2 = \{p,\ \neg p,\ p \vee s\}$, $S_3 = \emptyset$.

The algorithm given in Figure 2 checks whether a given knowledge-base $(S, Exact)$ is stratified. If so, the algorithm produces stratifications, and allows to construct recovered sets by providing corresponding (minimal) mcems of $(S, Exact)$ (see Theorem 1 below).

[5] To simplify matters we shall take here the empty clause as identical with $\bot$ rather than with f (as the definition of $\vee$ actually dictates).

[6] Note the similarity between the the dilution process and the Gelfond–Lifschitz transformation [GL88] used for providing semantics to logic programs with negations.

```
input: a knowledge-base KB = (S, Exact)
call RECOVER(S, ∅, 0)

procedure RECOVER(S, ν, i)
/* S – the i-th stratification level of KB,  ν – the valuation constructed so far. */
{
    if (S = ∅) then output ν and return;                          /* ν∈Ω(KB) */
    pos := {p∈A(S) | p∈S};                                        /* positive-facts */
    neg := {p∈A(S) | ¬p∈S};                                       /* negative-facts */
    if (pos = ∅ ∧ neg = ∅)  halt;                                 /* KB is not stratified */
    if (⊥ ∈ S)  return;                                  /* backtracking; not a stratification */
    if (∃p ∈ Exact ∩ pos ∩ neg)  return;                  /* backtracking; not a stratification */

    while ((∃p ∈ Exact ∩ pos) ∨ (∃p ∈ Exact ∩ neg) ∨ (∃p ∈ pos ∩ neg))  {
        pick such a p;
        if (p ∈ Exact ∩ pos)  {
            pos := pos \ {p};
            ν_i := {p : t};
        }
        if (p ∈ Exact ∩ neg)  {
            neg := neg \ {p};
            ν_i := {p : f};
        }
        else  {
            pos := pos \ {p};
            neg := neg \ {p};
            ν_i := {p : ⊤};
        }
        S_{i+1} := S_{ν_i};                                                  /* dilution */
        do (∀q s.t. ν_i(q) is undefined and q ∈ A(S) \ A(S_{i+1}))            /* filling */
            if (q ∉ Exact) then ν_i := ν_i ∪ {q : ⊥} else ν_i := ν_i ∪ {q : t};
        RECOVER(S_{i+1}, ν ∪ ν_i, i+1);
    }

    while (∃p ∈ pos ∪ neg)  {
        pick such a p;
        if (p ∈ pos)  {
            pos := pos \ {p};
            ν_i := {p : t};
        }
        else  {
            neg := neg \ {p};
            ν_i := {p : f};
        }
        S_{i+1} := S_{ν_i};                                                  /* dilution */
        do (∀q s.t. ν_i(q) is undefined and q ∈ A(S) \ A(S_{i+1}))            /* filling */
            if (q ∉ Exact) then ν_i := ν_i ∪ {q : ⊥} else ν_i := ν_i ∪ {q : t};
        RECOVER(S_{i+1}, ν ∪ ν_i, i+1);
    }
}
```

Fig. 2. An algorithm for recovering stratified knowledge-bases

Every valuation ν produced by the algorithm is determined by a sequence of picked atoms $p_0, p_1, \ldots, p_n$ of the calls to *RECOVER*. For shortening notations we shall just write ν instead of $\nu(p_0, p_1, \ldots, p_n)$.

Example 6. In our canonical example (3, 4, and 5), the algorithm produces two (minimal) mcems of KB: $M1 = \{p{:}t,\ q{:}t,\ r{:}\top,\ s{:}\bot,\ e{:}t\}$ and $M2 = \{p{:}\top,\ q{:}t,\ r{:}f,\ s{:}\bot,\ e{:}t\}$ Figure 3 illustrates the processing of the algorithm in this case.

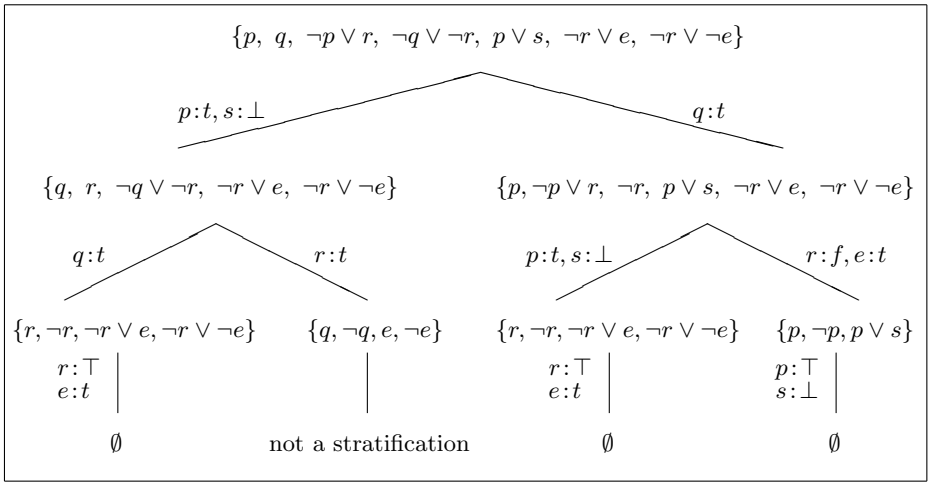

Fig. 3. Generation of minimal mcems and recovered sets for KB

Proposition 7. Let $KB = (S, Exact)$ be a finite knowledge-base. If it is stratified then the algorithm of Figure 2 finds every stratification of KB and outputs corresponding well-defined valuations for $\mathcal{A}(S)$. The algorithm halts without giving any valuation iff KB is not stratified.

Outline of proof: Every stratification of $(S, Exact)$ is produced by the algorithm since it performs a breadth first search on the atomic facts of every stratification level. The other parts of the proposition are easily verified, using the following facts:
(a) If a knowledge-base is stratified, then any order in which the facts are chosen determines stratification. This is so since dilution does not change facts; A fact (positive, negative, or both) of a certain level remains a fact in the successive levels unless it is used for the next dilution.
(b) The order in which the facts are chosen might be significant for checking condition (b) in the definition of stratification (Definition 11); This is the case, e.g., in our canonical example (see Figure 3). □

It follows from Proposition 7 that the algorithm halts with a valuation for a finite KB iff KB is stratified. For the rest of this section suppose, then, that KB is finite and stratified.

Theorem 1. Let ν be a valuation produced by the algorithm for a knowledge-base KB. Then: (a) $\nu \in \mathrm{kmin}(KB)$, (b) $\nu \in \mathrm{mcem}(KB)$, and (c) $\nu \in \Omega(KB)$.

Proof: We show the claim using three lemmas:

Lemma 1a: Every valuation ν produced by the algorithm is an exact model of KB.
Proof: Let ψ be a clause that appears in S. From Definition 9 and the algorithm of Figure 2 it is obvious that some part of ψ is eliminated from some S_{i+1} during the dilution of S_i. This happens iff (at least) one of its literals l is assigned a designated truth value by ν (note that a formula cannot be eliminated by sequently removing every literal of it according to (2) of Definition 9, since the last literal left must be assigned a designated value). By Lemma 2, then, $\nu(\psi) \in \{t, \top\}$, and so ν is a model of KB. ν is an *exact* model of KB, since no element of $Exact$ is assigned $\top$ or $\bot$ by the algorithm.

Lemma 1b: Every valuation produced by the algorithm is an mcem of KB.
Proof: The proof is by an induction on the number of the recursive steps (n) that are required for creating a valuation ν. If $n = 0$ then $S_1 = \emptyset$, so there is only the initial step in which ν might assign $\top$ only to a literal l that is both a positive and a negative fact of S. Since in this case l is assigned $\top$ by *every* model of S, ν must be most consistent. Suppose now that it takes $n \geq 1$ recursive steps to create ν. Denote by ν_i the part of the valuation ν that is determined during step i. Then:

$$(1): \quad Inc_\nu(S) = \bigcup_{0 \leq i \leq n} Inc_{\nu_i}(S_i) = Inc_{\nu_0}(S) \cup Inc_{\nu'}(S_1)$$

where $\nu' = \bigcup_{1 \leq i \leq n} \nu_i$. Now, let M be an exact model of KB. We show that $M \not\succ_{con} \nu$. For this suppose that M_1 is the reduction of M to $\mathcal{A}(S_1)$.

$$\begin{aligned}(2): \quad Inc_M(S) &= \{p \in \mathcal{A}(S) \setminus \mathcal{A}(S_1) \mid M(p) = \top\} \cup \{p \in \mathcal{A}(S_1) \mid M(p) = \top\} \\ &= \{p \in \mathcal{A}(S) \setminus \mathcal{A}(S_1) \mid M(p) = \top\} \cup Inc_{M_1}(S_1)\end{aligned}$$

By its definition, ν_0 might assign $\top$ only to $l \in \mathcal{L}(S)$ s.t. $l, \bar{l} \in S$. Obviously, such an l must be assigned $\top$ by every model of S, in particular $M(l) = \top$. Thus:

$$(3): \quad Inc_{\nu_0}(S) \subseteq \{p \in \mathcal{A}(S) \setminus \mathcal{A}(S_1) \mid M(p) = \top\}$$

• Suppose first that M_1 is an exact model of S_1. Since the creation of ν' requires only $n-1$ steps, then by the induction hypothesis ν' is an mcem of S_1. In particular, either $Inc_{\nu'}(S_1)$ and $Inc_{M_1}(S_1)$ are incomparable w.r.t. the containment relation, or else:

$$(4): \quad Inc_{\nu'}(S_1) \subseteq Inc_{M_1}(S_1)$$

From (1) – (4), either $Inc_\nu(S)$ and $Inc_M(S)$ are incomparable, or $Inc_\nu(S) \subseteq$

$Inc_M(S)$.

• If M_1 is *not* an exact model of S_1 then M_1 is cannot be a model of S_1 either, since it is a reduction of an exact model (M) of S. Thus there is a $\psi_1 \in S_1$ s.t. $M_1(\psi_1) \notin \{t, \top\}$. Since M is a model of S, then by Lemma 2 there is a $\psi \in S$ and $l \in \mathcal{L}(\psi)$ s.t. $M(l) \in \{t, \top\}$, and $\{l\} \cup \mathcal{L}(\psi_1) \subseteq \mathcal{L}(\psi)$. Obviously, $l \in \mathcal{A}(S) \setminus \mathcal{A}(S_1)$. But then $\nu_0(l) \notin \{t, \top\}$ (otherwise ψ is eliminated in the dilution of S, and so $\psi_1 \notin S_1$). Moreover, $\nu_0(\bar{l}) \in \{t, \top\}$, since if $\nu_0(\bar{l}) \notin \{t, \top\}$ then necessarily $\nu_0(l) = \bot$, and this happens only if ψ is eliminated in the dilution of S, i.e. $\psi_1 \notin S_1$. Therefore, $\nu_0(l) \notin \{t, \top\}$ and $\nu_0(\bar{l}) \in \{t, \top\}$, so $\nu_0(l) = f$. l is not assigned this value in the filling process, since again, this would imply that ψ is eliminated in the dilution of S, and so $\psi_1 \notin S_1$. Thus, by the definition of ν_0 and since S is stratified then necessarily $\bar{l} \in S$ and $l \notin S$. Hence $KB \models \bar{l}$. But M is an exact model of KB and so $M(\bar{l}) \in \{t, \top\}$. Since we have shown that $M(l) \in \{t, \top\}$ as well, it follows that $M(l) = \top$ while $\nu(l) = f$. Therefore $Inc_M(S) \not\subseteq Inc_\nu(S)$ in this case as well.

Lemma 1c: The algorithm produces k-minimal exact models of KB.
Proof: Again, we denote by ν_i the part of the valuation ν that is created in the i-th recursive call to *RECOVER*. The proof is by an induction on the number of recursive steps required to create ν:
$n = 0 : \nu_0$ may assign $\top$ only to a literal l s.t. $l \in S$ and $\bar{l} \in S$. In this case $\top$ is the only possible value for l, and so it is k-minimal. The same argument is true for any literal l s.t. $l \in S$ and $\bar{l} \notin S$ (for that l, $\nu(l) = t$). It is also obviously true for all the literals in $Exact$, and for the literals that are assigned $\bot$.
$n \geq 1$: Let M be a model of KB. We show that $M \not<_k \nu$. Let M_1 be the reduction of M to $\mathcal{A}(S_1)$, and suppose first that M_1 is an exact model of S_1. By the induction hypothesis ν_1 is a k-minimal exact model of S_1, thus there exists $p \in \mathcal{A}(S_1)$, s.t. $M_1(p) \not\leq_k \nu_1(p)$, therefore $M \not<_k \nu$. The other possibility is that M_1 is not an exact model of S_1. In this case M_1 cannot be a model of S_1 either, therefore there must be a clause $\psi_1 \in S_1$ s.t. $M_1(\psi_1) \notin \{t, \top\}$. Since M is an exact model of S, then by Lemma 2 there is a $\psi \in S$ and an $l \in \mathcal{L}(\psi)$ s.t. $M(l) \in \{t, \top\}$, and $\{l\} \cup \mathcal{L}(\psi_1) \subseteq \mathcal{L}(\psi)$. But then $\nu(l) \notin \{t, \top\}$ (Otherwise, ψ is eliminated in the dilution of S and so $\psi_1 \notin S_1$), while $M(l) \in \{t, \top\}$. It follows that $M(l) \not<_k \nu(l)$, therefore again $M \not<_k \nu$.

Now, by Lemma 1b, $\nu \in \text{mcem}(KB)$, by Lemma 1c, $\nu \in \text{kmin}(KB)$, and by both, $\nu \in \Omega(KB)$. This ends the proof of Theorem 1. □

Note: It is possible to assign any other truth value to the atoms that are assigned $\bot$, and still ν would be an exact model of KB. However, in such a case ν cannot be minimal w.r.t. $\leq_k$. Also, when assigning $\top$ instead of $\bot$, ν cannot be an mcem of KB. It is also possible to assign f to the elements of $Exact$ that are assigned t during the filling process without losing any of the properties discussed above.

Theorem 2. Let ν be a valuation produced by the algorithm for KB. Then KB_ν is a recovered set of KB.

Example 7. Consider again Example 6 and Figure 3. $KB_{M1} = \{p,\ q,\ p \vee s\}$ and $KB_{M2} = \{q,\ \neg q \vee \neg r,\ \neg r \vee e,\ \neg r \vee \neg e\}$ are the recovered sets of KB.

Proof of Theorem 2: By Theorem 1, every valuation ν that is generated by the algorithm is an exact model of KB. Thus, by Propositions 2, KB_ν is consistent in KB. It is left to show that KB_ν is also a *maximal* subset with this property. Suppose not. Then by Proposition 3 there is an mcem M of KB s.t. $KB_\nu \subset KB_M$, hence $Inc_\nu(KB) \neq Inc_M(KB)$. Since ν is also an mcem of KB (Theorem 1 again), there is a $p \in \mathcal{A}(S)$ s.t. $\nu(p) \neq \top$ while $M(p) = \top$. In particular, since M is an exact model of KB, $p \notin Exact$. Consider some $\psi \in S$ s.t. $p \in \mathcal{A}(\psi)$. Since $\psi \notin KB_M$, $\psi \notin KB_\nu$ either. Thus there is a $q \in \mathcal{A}(\psi)$ s.t. $\nu(q) = \top$. By the definition of ν this is possible only if there is a stratification $S_0, \ldots, S_n$ of S and an index $1 \leq i \leq n$ s.t. $q, \neg q \in S_i$. Therefore $\nu(p) \neq \bot$ (Otherwise, p as well as all the other elements of $\mathcal{A}(\psi)$ are diluted from S_j for some $j \leq i$, and so $q \notin \mathcal{A}(S_i)$). It follows that either $\nu(p) = t$ or $\nu(p) = f$. Since $p \notin Exact$, then by the construction of ν, either p or $\neg p$ is a strict fact (positive or negative) of some stratification level S_k of S. It follows that there is some $\phi \in S$ s.t. $p \in \mathcal{A}(\phi)$ and $\mathcal{A}(\phi) \cap Inc_\nu(S) = \emptyset$ (Otherwise, if there is some $r \in \mathcal{A}(\phi)$ s.t. $\nu(r) = \top$, then ϕ and its atoms are diluted in some stage before stage k, and so p cannot be a strict fact of S_k). Therefore $\phi \in KB_\nu$ while $\phi \notin KB_M$ – a contradiction. □

Finally, let's consider some complexity issues. As we have noted before, the problem of recovering arbitrary knowledge-base is at least NP-complete. Denote by $O(A^B)$ that it takes $O(A)$ running time to solve a certain problem when using an oracle for solving problems with complexity $O(B)$. Then our algorithm requires $O(|S|^{|\mathcal{A}(S)|})$ running time to recover a knowledge-base $(S, Exact)$ that is stratified.[7] As the following proposition shows, the complexity of the algorithm might sometimes be considerably reduced:

Proposition 8. Whenever each stratification level of $KB = (S, Exact)$ does not contain a pair of complementary exact literals, it takes only $O(|S| \cdot |\mathcal{A}(S)|)$ running time to check whether KB is stratified, and if so, this is also the time needed to find some recovered set of it.

Proof: By the conditions of the proposition, in order to find some recovered set of KB it is sufficient to execute the algorithm on a single sequence of recursive calls to *RECOVER*, without backtracking. Now, computing stage i of the recursion requires only $O(|S_i|)$ running time. Since there are at most $|\mathcal{A}(S)|$ recursive calls to *RECOVER*, the whole process does not take more than $O(|S| \cdot |\mathcal{A}(S)|)$ running time. By 2, this is also the time required to supply a recovered set KB_ν for KB. □

Obvious cases in which the condition of the last proposition is met are when $Exact = \emptyset$, or if there is no $l \in Exact$ s.t. both $l \in \mathcal{L}(S)$ and $\bar{l} \in \mathcal{L}(S)$.

[7] In our case, at every stratification level the oracle chooses a fact that yields, eventually, a stratification.

5 Model-based diagnosis

Suppose that we are given a description of some faulty device. Our goal is to find some minimal set of components the collective failure of which can explain an observed malfunction. In this section we show that the mcems and their corresponding recovered sets of the knowledge-base that describes that device are good candidates for providing accurate diagnoses regarding the faulty components. For that we first expand the discussion to first-order logic. It is possible to do so in a straightforward way, provided that each clause that contains variables is considered as universally quantified. Consequently, a knowledge-base containing non-grounded formula, ψ, will be viewed as representing the corresponding set of ground formulae formed by substituting each variable that appears in ψ with every possible element of the Herbrand universe, U. Formally: $KB^U = (S^U, Exact)$, where $S^U = \{\rho(\psi) \mid \psi \in S,\ \rho : var(\psi) \rightarrow U\}$, ρ is a *ground substitution* from the variables of every $\psi \in KB$ to the individuals of U, and $Exact$ consists of predicates every instance of which should be assigned classical values. The exact models are the elements of $mod(S^U, Exact) = \{M \in mod(S^U) \mid \forall p \in Exact\ \forall x_i \in U\ M(p(x_1, \ldots, x_n)) \in \{t, f\}\}$.

Example 8. Figure 4 depicts a binary full adder. It consists of five components: two and-gates A_1 and A_2, two xor-gates X_1 and X_2, and an or-gate O_1.

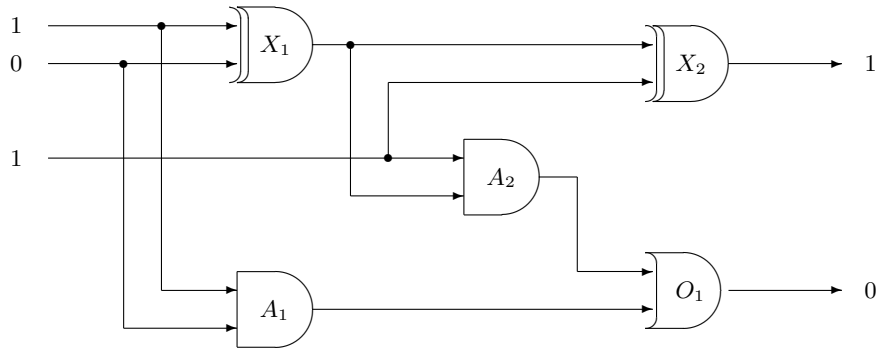

Fig. 4. A full adder

The full adder's description is given by system *FA* in Figure 5. Notice that this specific circuit is faulty; both circuit outputs are wrong for the given inputs.

The predicates $in1(x)$, $in2(x)$, and $out(x)$ of *FA* are assigned values that correspond to binary values of the wires of the system, therefore they should have only classical values. Also, it seems natural to restrict the values of the predicates *andGate*, *orGate*, and *xorGate* to be only classical as well. This is because we know in advance what is the kind of each gate G in the system, and so the only open question about G is whether it behaves as expected.

$andGate(x) \wedge ok(x) \rightarrow (out(x) \leftrightarrow (in1(x) \wedge in2(x)))$,
$xorGate(x) \wedge ok(x) \rightarrow (out(x) \leftrightarrow (in1(x) \oplus in2(x)))$,
$orGate(x) \wedge ok(x) \rightarrow (out(x) \leftrightarrow (in1(x) \vee in2(x)))$,
$andGate(x) \rightarrow (\neg orGate(x) \wedge \neg xorGate(x))$,
$xorGate(x) \rightarrow (\neg andGate(x) \wedge \neg orGate(x))$,
$orGate(x) \rightarrow (\neg andGate(x) \wedge \neg xorGate(x))$,
$in1(X_1) \leftrightarrow in1(A_1)$, $in2(X_1) \leftrightarrow in2(A_1)$, $in1(A_2) \leftrightarrow in2(X_2)$,
$out(X_1) \leftrightarrow in2(A_2)$, $out(X_1) \leftrightarrow in1(X_2)$, $out(A_1) \leftrightarrow in2(O_1)$, $out(A_2) \leftrightarrow in1(O_1)$,
$andGate(A_1)$, $andGate(A_2)$, $xorGate(X_1)$, $xorGate(X_2)$, $orGate(O_2)$,
$ok(A_1)$, $ok(A_2)$, $ok(X_1)$, $ok(X_2)$, $ok(O_1)$,
$in1(X_1)$, $\neg in2(X_1)$, $in1(A_2)$, $out(X_2)$, $\neg out(O_1)$

Fig. 5. The system *FA*

The knowledge-base that represents the full adder is then $(FA, Exact)$, where $Exact = \{in1, in2, out, andGate, orGate, xorGate\}$.

The table of Figure 6 lists the elements of mcem$(FA, Exact)$. We have omitted from the table predicates that have the same value in every exact model, and any predicate that has the same values as some predicate in the table.

Model No.	$in1$ $X2$	$in1$ $O1$	$in2$ $O1$	ok $A1$	ok $A2$	ok $X1$	ok $X2$	ok $O1$
$M1$	f	f	f	t	t	$\top$	t	t
$M2$	t	f	f	t	$\top$	t	$\top$	t
$M3$	t	t	f	t	t	t	$\top$	$\top$

Fig. 6. The mcems of $(FA, Exact)$

The mcems of $(FA, Exact)$, and the recovered sets that are associated with them preserve what Reiter [Re87] calls *the principle of parsimony*; they represent the conjecture that some minimal set of components are faulty. For instance, according to $M1$ the only component that behaves incorrectly is the xor gate X_1. The set that is associated with $M1$ reflects this indication:
$FA_{M1} = FA \backslash \{ok(X_1), xorGate(X_1) \wedge ok(X_1) \rightarrow (out(X_1) \leftrightarrow (in1(X_1) \oplus in2(X_1)))\}$
In particular, FA_{M_1} entails (w.r.t. both $\models$ and $\models_{mcem}$) $ok(x)$ for $x \in \{A_1, A_2, X_2, O_1\}$, but it does *not* entail $ok(X_1)$. Similarly, the other two mcems $M2$ and $M3$, together with their associated sets represent (respectively) situations, in which gates $\{X_2, A_2\}$ and gates $\{X_2, O_1\}$ are faulty. These are the generally accepted diagnoses of this case (see, e.g., [Re87, Example 2.2], [Gi88, Sections 15,16], and [Ra92, Examples 1,4]).

One might treat FA_{M1} as the preferred recovered set, since it is the only set that entails that only a single component is faulty, and one normally expects components to fail independently of each other. This kind of diagnosis is known as a *single fault diagnosis*.

As it is proved below, the correspondence in the previous example between the fault diagnoses and the inconsistent assignments of the mcems is not accidental. For showing that we first present two basic notions from the literature on model-based diagnosis:

Definition 12. [Re87] A *system* is a triple $(Sd, Comps, Obs)$, where: Sd, the *system description*, is a set of first order sentences; $Comps$, the *system components*, is a finite set of constants; and Obs, the *observations set*, is a finite set of sentences.

Definition 13. [Re87] A *diagnosis* is a minimal set $\Delta \subseteq Comps$ s.t. $Sd \cup Obs \cup \{ok(c) \mid c \in Comps \setminus \Delta\} \cup \{\neg ok(c) \mid c \in \Delta\}$ is classically consistent.

Definition 14. A *correct behavior assumption* for a given set of components $\Delta \subseteq Comps$ is the set $CBA(\Delta) = \{ok(c) \mid c \in \Delta\}$.

Definition 15. For a given system $(Sd, Comps, Obs)$, and a set of components $\Delta \subseteq Comps$, denote $S(\Delta) = Sd \cup Obs \cup CBA(\Delta)$. Whenever $\Delta = Comps$ we shall write just S instead of $S(Comps)$. We shall continue to assume that $S(\Delta)$ is a set of clauses.

Proposition 9. [Re87] Denote by $\models_{cl}$ the consequence relation of the first order classical logic.
a) Δ is a diagnosis for $(Sd, Comps, Obs)$ iff Δ is a minimal set s.t. $S(Comps \setminus \Delta)$ is classically consistent.
b) If Δ is a diagnosis for $(Sd, Comps, Obs)$ then $S(Comps \setminus \Delta) \models_{cl} \neg ok(c)$ for every $c \in \Delta$.

In the present treatment, unlike in the classical case, an inconsistency does not yield trivial reasoning, and only a subset of the atomic formulae must have classical values. In our terms, then, a diagnostic system is defined as follows:

Definition 16. A *diagnostic knowledge-base* is a knowledge-base $(S, Exact)$, where $S = Sd \cup Obs \cup CBA(Comps)$, and $Exact$ consists of every ground atom of S except the elements of $CBA(Comps)$.[8]

Theorem 3. Let $(S, Exact)$ be a diagnostic knowledge-base. An exact model M of $(S, Exact)$ is an mcem of $(S, Exact)$ iff $Inc_M(S) = CBA(\Delta)$ for some diagnosis Δ of S.

[8] Note that this requirement is met in Example 8.

Proof: ($\Leftarrow$) Assume that M is an exact model of $(S, Exact)$ and that Δ is a diagnosis of S s.t. $Inc_M(S) = CBA(\Delta)$. If M is not an mcem of S then by Proposition 1 there is an exact model M' s.t. $Inc_{M'}(S) \subset Inc_M(S) = CBA(\Delta)$, i.e.: there is a $c_0 \in \Delta$ s.t. $M'(ok(c_0)) \neq \top$. But:
(a) M' is a model of S and $ok(c_0) \in S$ thus $M'(ok(c_0)) \in \{t, \top\}$, and
(b) By Proposition 9(b), $S(Comps \setminus \Delta) \models_{cl} \neg ok(c_0)$. Hence by Lemma 4.11 of [AA96] [9], $S(Comps \setminus \Delta) \models_{mcem} \neg ok(c_0)$. Since M is a (most) consistent exact model of $S(Comps \setminus \Delta)$, so is M'. Therefore $M'(\neg ok(c_0)) \in \{t, \top\}$.
By (a) and (b), $M'(ok(c_0)) = \top$ – a contradiction.
($\Rightarrow$) From the condition on $Exact$ it follows that for every exact model M of $(S, Exact)$, $Inc_M(S) \subseteq CBA(Comps)$. Suppose, then, that M is an mcem of $(S, Exact)$ and that $Inc_M(S) = CBA(\Delta)$ for some $\Delta \subseteq Comp$. By Proposition 9, in order to prove that Δ is a diagnosis for S it is sufficient to show that Δ is a minimal set such that $S(Comps \setminus \Delta)$ is classically consistent. Suppose not. Then there is a proper subset $\Delta' \subset \Delta$ s.t. $S(Comps \setminus \Delta')$ is classically consistent, and so has a consistent model, N. Let M' be the following valuation:

$$M'(p) = \begin{cases} N(p) & \text{if } p \in \mathcal{A}(S(Comps \setminus \Delta')). \\ \top & \text{otherwise.} \end{cases}$$

It is easy to verify (using Lemma 2) that M' is a model of S. Therefore, since $Exact(S) \subset \mathcal{A}(S(Comps \setminus \Delta'))$, M' is an exact model of $mod(S, Exact)$. Moreover, $Inc_{M'}(S) = CBA(\Delta')$, and $\Delta' \subset \Delta$, thus $Inc_{M'}(S) = CBA(\Delta') \subset CBA(\Delta) = Inc_M(S)$. It follows that M cannot be a mcem of $(S, Exact)$. □

Corollary 1. Let $(S, Exact)$ be a diagnostic knowledge-base. If Δ is a diagnosis of S then there exists an mcem M of $(S, Exact)$ s.t. $Inc_M(S) = CBA(\Delta)$.

Proof: By Proposition 9(a), $S(Comps \setminus \Delta)$ is classically consistent, therefore there is an exact model M of $(S, Exact)$ that assigns $\top$ only to $CBA(\Delta)$. This M is an mcem of $(S, Exact)$ by Theorem 3. □

It follows that whenever the requirement of Theorem 3 is met and $(S, Exact)$ is stratified, one can use the algorithm of Section 4 for finding diagnoses and constructing recovered knowledge-bases of the faulty system. This is the case, e.g., in Example 8.

6 Conclusion

We have proposed a four-valued mechanism for recovering consistent data from inconsistent set of assertions. This approach regards some contradictory data as useless, and considers all the remaining information as unaffected. The logics

[9] According to that lemma, if S is a classically consistent set of assertions, ψ is a clause that does not contain any pair of complementary literals, and ψ classically follows from S, then $S \models_{mcem} \psi$. In [AA96] this is proved only for the case $Exact = \emptyset$, but the proof can easily be adapted to the general case.

behind this kind of method are nonmonotonic and paraconsistent in nature. For a common family of knowledge-bases we have also provided an algorithm for an automatic recovery. Our method is particularly useful for diagnostic systems, where it might be used for supplying a description of the well-behaved parts of a faulty device.

References

[AA94] O.Arieli, A.Avron. *Logical bilattices and inconsistent data.* Proc. of the 9th IEEE Annual Symp. on Logic in Computer Science (LICS'94). IEEE Press, pp.468-476; 1994.

[AA95] O.Arieli, A.Avron. *A bilattice-based approach to recover consistent data from inconsistent knowledge-bases.* Proc. of the 4th Bar-Ilan Symp. on Foundations of Artificial Intelligence (BISFAI'95), pp. 231-240; 1995.

[AA96] O.Arieli, A.Avron. *Reasoning with logical bilattices.* Journal of Logic, Language, and Information, Vol.5, pp. 25-63; 1996

[Be77a] N.D.Belnap. *A useful four-valued logic.* Modern Uses of Multiple-Valued Logic (G.Epstein, J.M.Dunn - Eds.), Reidel, Dordrecht, pp. 7-37; 1977.

[Be77b] N.D.Belnap. *How computer should think.* Contemporary Aspects of Philosophy (G.Ryle - Ed.), Oriel Press, Stocksfield, England, pp. 30-56; 1977.

[dC74] N.C.A.da-Costa. *On the theory of inconsistent formal systems.* Notre Damm Journal of Formal Logic, Vol.15, pp. 497-510; 1974.

[Gi88] M.L.Ginsberg. *Multivalued logics: A uniform approach to reasoning in AI.* Computer Intelligence, Vol.4, pp.256-316; 1988.

[GL88] M.Gelfond, V.Lifschitz. *The stable model semantics for logic programming.* Proc. of the 5th logic programming symposium MIT Press, Cambridge, MA, pp.1070-1080; 1988.

[KLM90] S.Kraus, D.Lehmann, M.Magidor. *Nonmonotonic reasoning, preferential models and cumulative logics.* Journal of Artificial Intelligence, Vol.44, No.1-2, pp. 167-207; 1990.

[KL92] M.Kifer, E.L.Lozinskii. *A logic for reasoning with inconsistency.* Journal of Automated reasoning. Vol.9, No.2, pp. 179-215; 1992.

[LM92] D.Lehmann, M.Magidor. *What does a conditional knowledge base entail?*. Journal of Artificial Intelligence, Vol.55, pp. 1-60; 1992.

[Pr91] G.Priest. *Minimally inconsistent LP.* Studia Logica, Vol.50, pp. 321-331; 1991.

[Ra92] O.Raiman. *The alibi principle.* Readings in Model-Based Diagnosis (W.Hamscher, L.Console, J.de-Kleer - Eds.), pp. 66-70; 1992.

[Re87] R.Reiter. *A theory of diagnosis from first principles.* Journal of Artificial Intelligence, Vol.32, No.1, pp. 57-95; 1987.

[Sh87] Y.Shoham. *A semantical approach to nonmonotonic logics.* Readings in Non-Monotonic Reasoning (M.L.Ginsberg – ed.), Los-Altos, CA, pp.227-249; 1987.

[Sh88] Y.Shoham. *Reasoning about change: time and causation from the standpoint of artificial intelligence*. MIT Press; 1988.