A Bilattice-based Approach to Recover Consistent Data from Inconsistent Knowledge-Bases

Ofer Arieli
Department of Computer Science
School of Mathematical Sciences
Tel-Aviv University
Ramat-Aviv 69978, ISRAEL.
Email: ofera@math.tau.ac.il

Arnon Avron
Department of Computer Science
School of Mathematical Sciences
Tel-Aviv University
Ramat-Aviv 69978, ISRAEL.
Email: aa@math.tau.ac.il

Abstract

Bilattices, which have been shown to be very useful in logic programming, are used here for recovering consistent data from an inconsistent knowledge-base. Our method is conservative in the sense that it considers the contradictory data as useless, and regards all the remaining information unaffected. This kind of approach is nonmonotonic and paraconsistent in nature.

1 Introduction

One of the most significant drawbacks of the classical calculus is its inability to deduce nontrivial results in the presence of an inconsistency. The warranty of drawing any conclusion whatsoever just because of the existence of a contradiction, is certainly unrealistic. Nevertheless, the classical calculus is still a very convenient framework to work with; adding new mechanisms or connectives to it generally cause a considerable growth in the computational complexity needed to maintain the resulting system. The purpose of this work, then, is to propose means that would allow drawing conclusions from systems that are based on classical logic, although the information might become temporarily inconsistent.

The scenario we think of is the following: Suppose that a given first-order classical knowledge-base has become inconsistent. For the reason stated above, any attempt to deduce meaningful inferences from this "polluted" knowledge-base is useless. Our approach solves this "dead-end" by temporarily expanding the semantics to be multi-valued, and this without changing the knowledge-base syntactically.

How do we practically recover consistent data from an inconsistent knowledge-base without changing it? The first step, as we have implied, is to switch into a special multi-valued framework. For this, we use a special algebraic structures called bilattices. Bilattices were first proposed by Ginsberg (see [Gi88]) as a basis for a general framework for many applications. This notion was further developed by Fitting ([Fi90a, Fi94]), who showed that bilattices are most suitable for logic programming ([Fi89, Fi90b, Fi91, Fi93]). In bilattices the elements (which are also referred to as "truth values") are simultaneously arranged in two partial orders: one, \( \leq_t \), may intuitively be understood as a measure of the degree of truth that each element represents; the other, \( \leq_k \), describes (again, intuitively) differences in the amount of knowledge (or information) that each element exhibits on the assertions that it is supposed to represent.

The next step is to develop a mechanism that enables paraconsistent inferences. For this, we use an epistemic entailment proposed in [KL92] as the consequence relation of the logic \(^1\). This relation can be viewed as a kind of a "closed word assumption", since it considers only the "most consistent" models.

\(^1\)See [AA94] for a detailed discussion on the advantages of the present logic with respect to the logic of [KL92] in particular, and to annotated logics [Su90a, Su90b, KS92] in general.
operators would not play a central role in what follows, since we will be most interested in the "classical" operators \( \land \) and \( \lor \). However, our method allows the usage of these operators without any further effort, so we shall refer to them as well.

2 Preliminary definitions and notations

In this section we briefly review the notions that will be significant in what follows. For a more detailed presentation of the following notions, refer to [AA94b].

2.1 Bilattices

Definition 2.1 [Gi88] A bilattice is a structure \( B = (B, \leq_t, \leq_k, \land, \lor, \top, \bot) \) such that \( B \) is a non-empty set containing at least two elements; \( (B, \leq_t) \), \( (B, \leq_k) \) are complete lattices; and \( \bot \) is a unary operation on \( B \) s.t.: (a) if \( a \leq_t b \) then \( \neg a \geq_t \neg b \); (b) if \( a \leq_k b \) then \( \neg a \leq_k \neg b \); (c) \( \neg \neg a = a \).

Notation 2.2 Following Fitting, we shall use \( \land \) and \( \lor \) for the meet and join which correspond to \( \leq_t \), and \( \top, \bot \) for the meet and join under \( \leq_k \). He suggested to intuitively understand \( \land \) and \( \lor \) as representing the "consensus" and "accept all" operations, respectively. \( f \) and \( t \) will denote, respectively, the least and the greatest element w.r.t. \( \leq_t \), while \( \bot \) and \( \top \) - the least and the greatest element w.r.t. \( \leq_k \). While \( f \) and \( t \) may have their usual intuitive meaning, \( \bot \) and \( \top \) could be thought of as representing no information and inconsistent knowledge, respectively. Obviously, \( f, t, \bot \) and \( \top \) are all different (see lemma 2.3).

Lemma 2.3 [Gi88] Let \( B = (B, \leq_t, \leq_k, \land, \lor, \top, \bot) \) be a bilattice, and suppose that \( a, b \in B \).

(a) \( \neg(a \land b) = \neg a \lor \neg b \); \( \neg(a \lor b) = \neg a \land \neg b \); \( \neg(a \otimes b) = \neg a \otimes \neg b \); \( \neg(a \oplus b) = \neg a \oplus \neg b \).

(b) \( f = t \); \( t = f \); \( \bot = \top \); \( \top = \bot \).

Definition 2.4 [AA94a] Let \( B = (B, \leq_t, \leq_k, \land, \lor, \top, \bot) \) be a bilattice.

(a) A filter is a nonempty set \( F \subseteq B \) s.t: (i) \( a \land b \in F \) iff \( a \in F \) and \( b \in F \), (ii) \( a \land b \in F \) iff \( a \in F \) and \( b \in F \).

(b) A filter \( F \) is prime, if it also satisfies: (i) \( a \land b \in F \) iff \( a \in F \) or \( b \in F \), (ii) \( a \land b \in F \) iff \( a \in F \) or \( b \in F \).

Given a bilattice \( B \), it may contain many filters. The elements of a filter are taken to be designated truth values of \( B \); i.e., they represent formulae that are considered true.

Definition 2.5 Given a bilattice \( B = (B, \leq_t, \leq_k, \land, \lor, \top, \bot) \). Two natural candidates for being the set of the designated values of \( B \) are \( D_k(B) = \{ b \in B \mid b \geq_k t \} \) and \( D_t(B) = \{ b \in B \mid b \geq_t \top \} \).

Lemma 2.6 Let \( B = (B, \leq_t, \leq_k, \land, \lor, \top, \bot) \) be a bilattice. For every \( b \in B \), \( \{ b, \neg b \} \subseteq D_t(B) \) iff \( b = \top \).

Proof: \( \{ b, \neg b \} \subseteq D_t(B) \) if \( b \geq_t \top \) and \( \neg b \geq_t \top \) if \( b \geq_t \top \) and \( b \leq_t \top \) if \( b = \top \). \( \Box \)

Definition 2.7 [AA94a] A logical bilattice is a pair \( (B, F) \), where \( B \) is a bilattice, and \( F \) is a prime filter.

Example 2.8 Belnap's \textit{FOUR}, Ginsberg's \textit{DEFAULT}, and \textit{NINE} (figure 1) are all logical bilattices with \( F = D_k(\cdot) = D_t(\cdot) \). In case of \textit{FOUR} and \textit{DEFAULT} there is no other filter. \textit{NINE} induces also another logical bilattice, in which \( F = D_k(\text{NINE}) \cup \{ a f, d \top, dl \} \).

\[ \text{These operators would not play a central role in what follows, since we will be most interested in the "classical" operators } \land \text{ and } \lor. \text{ However, our method allows the usage of these operators without any further effort, so we shall refer to them as well.} \]
2.2 The logic

**Definition 2.9** Let $B$ be an arbitrary bilattice. The language $BL(B)$ (Bilattice-based Language over $B$) is the standard propositional language over $\{\land, \lor, \neg, \Box, \Diamond\}$ enriched with a propositional constant for each element in $B$. In what follows we shall fix $B$, so we shall shorten $BL(B)$ to $BL$.

**Definition 2.10** Let $KB$ be a set of formulae above $BL$.

a) $A(KB)$ denotes the set of the atomic formulae that appear in some formula of $KB$.

b) $L(KB)$ denotes the set of the literals that appear in some formula of $KB$.

d) A valuation $\nu$ is a function that assigns a truth value from $B$ to each atomic formula, and map every constant to its corresponding value in $B$. Any valuation is extended to complex formulas in the standard way. We shall sometimes denote $\psi: b \in \nu$ (or $\nu = \{\psi : b, \ldots\}$) instead of $\nu(\psi) = b$.

e) Given $(B, F)$, we will say that $\nu$ satisfies $\psi$ ($\nu \models \psi$), iff $\nu(\psi) \in F$.

f) A valuation that satisfies every formula in a given set of formulas, $KB$, is said to be a model of $KB$. The set of the models of $KB$ will be denoted $mod(KB)$.

The next notion describes the truth values of $B$ that represent inconsistent beliefs:

**Definition 2.12** [AA94a] Let $(B, F)$ be a logical bilattice. A subset $\mathcal{I}$ of $B$ is an inconsistency set, if it has the following properties: (a) $b \in \mathcal{I}$ iff $\neg b \in \mathcal{I}$, (b) $b \in F \cap \mathcal{I}$ iff $b \in F$ and $\neg b \in F$.

**Example 2.13** $\mathcal{I}_1 = \{b \mid b \in F \land \neg b \in F\}$ is the minimal inconsistency set in every logical bilattice. $\mathcal{I}_2 = \{b \mid b = \neg b\}$ is inconsistency set in FOUR, DEFAULT, and NINE with $F = D_k(\cdot)$.

In the following discussion we fix some logical bilattice $(B, F)$ as well as an inconsistency set $\mathcal{I}$ of it. $i \in L(KB)$ will denote an arbitrary literal, $\overline{i}$ - its complementary, and $p, q \in A(KB)$ - atomic formulae.

**Notation 2.14** Given a valuation $M$ on $KB$. Denote: $Inc_M(KB) = \{p \in A(KB) \mid M(p) \in \mathcal{I}\}$.

Note that by (b) of definition 2.12, always $\top \in \mathcal{I}$ and $\bot \notin \mathcal{I}$. Hence, by (a), $f \notin \mathcal{I}$.

Note that $\bot \notin \mathcal{I}_1$, while $\bot \in \mathcal{I}_2$. Indeed, one of the major considerations when choosing an inconsistency set, is whether to include $\bot$ in $\mathcal{I}$ or not. Although in every bilattice $\bot = \bot$ (see lemma 2.3), $\bot$ intuitively reflects no knowledge whatsoever about the assertion it represents; in particular one might not take such assertions as inconsistent.
Definition 2.15 Let $M, N$ be two models of a finite set of formulae, $KB$.

a) $M$ is more consistent than $N$ ($M \succ_{\text{con}} N$), iff $\text{Inc}_M(KB) \subset \text{Inc}_N(KB)$.

b) $M$ is a most consistent model of $KB$ (mcm, in short), if there is no other model of $KB$ which is more consistent than $M$. The set of all the mcms of $KB$ will be denoted by $\text{con}(KB)$.

c) $M$ is smaller than $N$ (with respect to $<_{\text{con}}$), $M <_{\text{con}} N$, if for any $p \in A(KB)$, $M(p) \leq_k N(p)$, and there is $q \in A(KB) \setminus M(q) <_{\text{con}} N(q)$.

d) $M$ is a minimal model of $KB$, if there is no other model of $KB$ which is smaller than $M$. The set of all the minimal models of $KB$ will be denoted by $\text{min}(KB)$.

Definition 2.16 Let $KB$ be a finite sets of formulae and $\psi$ - a formula. Let $S$ be any set of valuations. We denote $KB \models_s \psi$ if each model of $KB$ which is in $S$, is also a model of $\psi$.

Some particularly interesting instances of definition 2.16 are the following: (1) $KB \models_{\text{m od}(KB)} \psi$ if every model of $KB$ is a model of $\psi$ (abbreviation: $KB \models \psi$), (2) $KB \models_{\text{con}(KB)} \psi$ if every mcm of $KB$ is a model of $\psi$ (abbreviation: $KB \models_{\text{con}} \psi$), and (3) $KB \models_{\text{min}(KB)} \psi$ if every minimal model of $KB$ is a model of $\psi$ (abbreviation: $KB \models_{\text{min}} \psi$).

Example 2.17 Let $KB = \{s, \neg s, r_1, r_1 \rightarrow \neg r_2, r_2 \rightarrow d\}$ and $\mathcal{B} = \text{FOUR}$ with $\mathcal{F} = \{t, \top\}$. The models of $KB$ are listed in figure 2 below. It follows that $\text{con}(KB) = \{M_1, M_2, M_3\}$ provided that $\bot \notin \mathcal{I}$, while if $\bot \in \mathcal{I}$, $\text{con}(KB) = \{M_2, M_3\}$. Also, $\text{min}(KB) = \{M_1, M_9\}$, thus $KB \models_{\text{con}} \neg r_2$, while $KB \not\models \neg r_2$ and $KB \not\models_{\text{min}} \neg r_2$.

<table>
<thead>
<tr>
<th>Model No.</th>
<th>$s$</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_3$</td>
<td>T</td>
<td>T</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_4$</td>
<td>T</td>
<td>T</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_5$</td>
<td>T</td>
<td>T</td>
<td>$\bot$, $T$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_6$</td>
<td>T</td>
<td>T</td>
<td>$\bot$, $f$, $T$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_7$</td>
<td>T</td>
<td>T</td>
<td>$\bot$, $f$, $T$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_8$</td>
<td>T</td>
<td>T</td>
<td>$\bot$, $f$, $T$</td>
<td>$\bot$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model No.</th>
<th>$s$</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_9$</td>
<td>T</td>
<td>T</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{10}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{11}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{12}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{13}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{14}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{15}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{16}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{17}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{18}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{19}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$M_{20}$</td>
<td>T</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\bot$</td>
</tr>
</tbody>
</table>

Figure 2: The models of $KB$

2.3 The knowledge-bases

Definition 2.18 A formula $\psi$ over $BL$ is an extended clause, if $\psi$ is a literal, or $\psi = \phi \lor \varphi$, where $\phi$ and $\varphi$ are extended clauses, or $\psi = \phi \land \varphi$, where $\phi$ and $\varphi$ are extended clauses.

Definition 2.19 A formula $\psi$ is said to be normalized, if it has no subformula of the form $\phi \lor \varphi$, $\phi \land \varphi$, $\phi \lor \psi$, $\phi \land \psi$, or $\neg \varphi$.

Lemma 2.20 For every formula $\psi$ there is an equivalent normalized formula $\psi'$ such that for every valuation $\nu$, $\nu(\psi) \in \mathcal{F}$ if $\nu(\psi') \in \mathcal{F}$.

From now on, unless otherwise stated, the knowledge-bases that we shall consider are finite sets of normalized extended-clauses. As the next proposition shows, presenting the formulae in an (normalized) extended clause form does not reduce the generality:

Proposition 2.21 For every formula $\psi$ over $BL$ there is a set $S$ of normalized extended clauses such that for every valuation $M$, $M \models \psi$ if $M \models S$.

Proof: By an induction on the structure of the negation normal form of $\psi$. □

Lemma 2.22 Let $\psi$ be an extended clause over $BL$, $l_i$ $(i = 1 \ldots n)$ - its literals, and $\nu$ - a valuation on $A(\psi)$. Then $\nu \models \psi$ iff there is $1 \leq i \leq n$ s.t. $\nu \models l_i$.

Proof: By an induction on the structure of $\psi$. □


3 Classification of the atomic formulae

The first step to recover inconsistent situations is to identify the atomic formulae that are involved in the conflicts. In order to do so, we divide the atomic formulae that appear in the clauses of the knowledge-base into four subsets as follows:

**Definition 3.1** Let $KB$ be a set of formulae, and $i \in \mathcal{L}(KB)$.

- a) If $KB \models_{\text{con}} i$ and $KB \models_{\text{con}} \overline{i}$, then $i$ is said to be **spoiled**.
- b) If $KB \models_{\text{con}} i$ and $KB \not\models_{\text{con}} \overline{i}$, then $i$ is said to be **recoverable**.
- c) If $KB \not\models_{\text{con}} i$ and $KB \not\models_{\text{con}} \overline{i}$, then $i$ is said to be **damaged**.

**Example 3.2** In example 2.17, $s$ is spoiled, $r_1$ and $\overline{r_2}$ are recoverable, and $d$ is damaged.

3.1 The spoiled literals

We treat first those literals that form, as their name suggests, the “core” of the inconsistency in $KB$. As the following theorem suggests, those literals are very easy to detect:

**Theorem 3.3** Let $KB$ be a knowledge-base, and $l \in \mathcal{L}(KB)$. The following conditions are equivalent:

- (a) $l$ is a spoiled literal of $KB$.
- (b) $M(i) \in \mathcal{I} \cap \mathcal{F}$ for every model $M$ of $KB$.
- (c) $M'(i) \in \mathcal{I} \cap \mathcal{F}$ for every mcm $M'$ of $KB$.
- (d) $\{i, \overline{i}\} \subseteq KB$.

**Proof**: The only nontrivial transition in the sequence (a) $\rightarrow$ (c) $\rightarrow$ (d) $\rightarrow$ (b) $\rightarrow$ (a) is (c) $\rightarrow$ (d): Suppose that for every mcm $M'$ of $KB$, $M'(i) \in \mathcal{I} \cap \mathcal{F}$. Since $KB$ is finite, then for every model of $KB$ there is an mcm of $KB$ which is more consistent or equal to it. Hence $i$ is assigned some consistent truth value in every model of $KB$. Assume that $l \in \{p, \overline{p}\}$ for some $p \in A(KB)$, and consider the following valuations: $\nu_l = \{q: \top | q \in A(KB), q \neq p\} \cup \{p: T\}$, $\nu_f = \{q: \top | q \in A(KB), q \neq p\} \cup \{p: f\}$. Since $\nu_l$ is not a model of $KB$ (because $p$ has a consistent value under $\nu_l$), $\overline{p} \in KB$ (otherwise, every formula $\psi \in KB$ contains a literal $l'$ such that $\nu_l(l') \in \mathcal{F}$, and so $\nu_l \models \psi$ by lemma 2.22). Similarly, since $\nu_f$ is not a model of $KB$, $p \in KB$. □

**Corollary 3.4** If $F = D_l(B)$ then every model of $KB$ assigns $\top$ to every spoiled literal.

**Proof**: Immediate from (d) of theorem 3.3, since $\{b, \overline{b}\} \in D_l(B)$ iff $b = \top$ (see lemma 2.6). □

3.2 The recoverable literals and their support sets

The recoverable literals are those that may be viewed as the “robust” part of a given inconsistent knowledge-base, since all the mcm’s “agree” on their validity. As we shall see, each recoverable literal $i$ can be associated with a consistent subset, which preserves the information about $i$.

**Definition 3.5** Let $KB$ be a finite set of normalized extended clause in BL.

- a) A model $M$ of $KB$ is **consistent** if it assigns a consistent value to every atom that appears in $A(KB)$.
- b) $KB$ is **consistent** if it has a consistent model.
- c) A subset $KB' \subseteq KB$ is **consistent in $KB$** if $KB'$ is a consistent set, and (at least) one of its consistent models is expandable to a (not necessarily consistent) model of $KB$.

**Example 3.6** $KB' = \{q\}$ is a consistent set, which is **not** consistent in $KB = \{q, \overline{q}\}$, since there is no consistent model of $KB'$ that is expandable to a model of $KB$.

**Definition 3.7** A set of normalized extended clauses $SS(i)$ is a **support set** of $i$ (or: $SS(i)$ **supports** $i$), if $SS(i)$ satisfies the following conditions: (a) $SS(i) \subseteq KB$; $SS(i)$ is not empty, (b) $SS(i)$ is consistent in $KB$, (c) $SS(i) \models_{\text{con}} i$ and $SS(i) \not\models_{\text{con}} \overline{i}$.

---

^5As noted before, these notions (but not the context) are taken from [KL92].
**Definition 3.8** If \(SS(l)\) supports \(l\), and there is no support set \(SS'(l)\) s.t. \(SS(l) \subset SS'(l)\), then \(SS(l)\) is said to be a **maximal support set** of \(l\), or **recovered subset** of \(KB\). A knowledge-base that has a recovered subset is called **recoverable**.

**Example 3.9** Consider again the example given in 2.17 and 3.2: \(KB = \{s, \neg s, r_1, r_1 \leftarrow \neg r_2, r_2 \rightarrow d\}\). Here \(KB\) is a recoverable knowledge-base, since \(S = \{r_1, r_1 \leftarrow \neg r_2, r_2 \rightarrow d\}\) is a maximal support set of \(r_1\) and \(\neg r_2\). Note that \(S\) does not support \(d\), since \(S \not\models d\).

**Theorem 3.10** Every recoverable literal has a support set.

**Outline of proof:** Without loss of generality, suppose that \(l = p\), where \(p \in A(KB)\) is recoverable. Let \(M\) be an mcm of \(KB\), s.t. \(M(p) \in F \setminus I\). Suppose that \(\{r_1, r_2, \ldots, r_n\}\) are the members of \(\bar{A}(KB) \setminus Inc_M(KB)\). Define: \(SS'(p) = \{\psi \in KB \mid A(\psi) \subseteq \{r_1, \ldots, r_n\}\}\). This \(SS'(p)\) is a support set for \(p\). \(\Box\)

**Theorem 3.11** Let \(l\) be a recoverable literal in \(KB\), then no subset of \(KB\) supports \(\bar{l}\).

**Proof:** Without a loss of generality, suppose that \(l = p\), and assume that \(SS'(p)\) is a support set for \(\neg p\). \(SS'(p)\) has a consistent model, \(M'\), which is expandable to a model \(M\) of \(KB\). \(M\) preserves the valuations of \(M'\) on \(A(SS'(p))\), so in particular \(M(q) = M'(q) \not\in I\) for every \(q \in A(SS'(p))\). Let \(N\) be an mcm of \(KB\) s.t. \(N \supseteq \text{con} M\) (such an \(N\) exists since \(KB\) is finite). Since \(N \supseteq \text{con} M\) then still \(N(q) \not\in I\) for every \(q \in A(SS'(p))\). Also, \(N\) is an mcm of \(KB\) and \(p\) is a recoverable atom of \(KB\), hence \(N(p) \in F\). Let \(N'\) be the reduction of \(N\) to \(SS'(p)\). Since \(N'\) is identical to \(N\) on \(A(SS'(p))\), and since \(N\) is a model of \(KB\), then: (a) \(N'\) is a model of \(SS'(p)\), (b) \(N'(q) \not\in I\) for every \(q \in A(SS'(p))\), and (c) \(N'(p) \not\in F\). Form (a) and (b), then, \(N'\) is a consistent model of \(SS'(p)\), and so from (c), \(N'(-p) \not\in F\). Thus \(SS'(p) \not\models \neg p\); a contradiction. \(\Box\)

**Proposition 3.12** Let \(l\) be a literal s.t. \(KB \models \text{con} l\). \(l\) is recoverable iff it has a support set.

**Proof:** The “only if” part was proved in theorem 3.10. For the “if” direction note that since \(l\) has a support set, it cannot be spoiled. \(l\) cannot be damaged either, since \(KB \models \text{con} l\). This is also the reason why \(\bar{l}\) cannot be recoverable. The only possibility left, then, is that \(l\) is recoverable. \(\Box\)

**Corollary 3.13** Every literal \(l\) such that \(\{l\} \subseteq KB\) and \(\{\bar{l}\} \not\subseteq KB\) is recoverable \(^6\).

**Proof:** It is easy to see that if \(\{\bar{l}\} \not\subseteq KB\) then \(SS(l) = \{l\}\) is a support set of \(l\) (not necessarily maximal). Since \(SS(l) \models l\) and \(\models\) is monotonic, then \(KB \models l\) as well, and so \(KB \models \text{con} l\). By 3.12, \(l\) is recoverable. \(\Box\)

Another refinement of proposition 3.12 is the following: For \(KB' \subset KB\) denote by \(\text{con}(KB) \mid KB'\) the **reductions** of the mcms of \(KB\) to the language of \(KB'\). Then:

**Proposition 3.14** \(l\) is a recoverable literal iff it has a support set \(SS(l)\) s.t. \(SS(l) \models \text{con}(KB) \mid KB'\) \(l\).

**Proof:** If \(l\) is recoverable, then by theorem 3.10 it must have a support set \(SS(l)\). Also, since \(l\) is recoverable, it is assigned a designated truth value by every mcm. These values are kept when reducing the mcms to the language of \(SS(l)\), hence \(SS(l) \models \text{con}(KB) \mid SS(l)\). For the converse, it is easy to verify that \(l\) cannot be either spoiled or damaged, and that \(\bar{l}\) cannot be recoverable. \(\Box\)

When a recoverable literal has several support sets it seems reasonable to prefer those that are maximal (w.r.t. containment relation). We next consider such sets:

**Definition 3.15** Let \(l\) be a recoverable literal in \(KB\), and \(M - \text{an mcm of KB such that}\ M(l) \in F \setminus I\). The support set of \(l\) that is associated with \(M\) is: \(SS_M(l) = \{\psi \in KB \mid A(\psi) \cap \text{Inc}_M(KB) = \emptyset\}\) \(^7\).

\(^6\)The converse of corollary 3.13 is, of course, not true. Consider, e.g. \(KB = \{p \lor \neg p, p \land q, \neg p \land q\}\); \(q\) is recoverable although \(\{q\} \not\subseteq KB\). Moreover, this knowledge-base contains a recoverable literal although there is no \(l \in L(KB)\) s.t. \(\{l\} \subseteq KB\).

\(^7\)This is indeed a support set of \(l\). See the proof of theorem 3.10.
Proposition 3.16 Every maximal support sets of a recoverable literal $l$ is associated with some mcm $M$ of $KB$ s.t. $M(l) \not\in I$.

Proof: Suppose that $SS'(l)$ is an arbitrary support set of $l$. Let $N'$ be a consistent model of $SS'(l)$, and $N$ - its expansion to the whole $KB$. Consider any mcm $M$ that satisfies $N \leq_{con} M$. Since $A(SS'(l)) \subseteq A(KB) \setminus nec(KB) \subseteq A(KB) \setminus nec(M(KB))$, then each formula $\phi \in SS'(l)$ consists only of literals that are assigned consistent truth values under $M$. Hence $SS'(l) \subseteq SS_M(l)$. □

Corollary 3.17 A knowledge-base is recoverable iff it has a recoverable literal.

Proof: By definition 3.8, a recoverable knowledge-base $KB$ must have a maximal support set, and by proposition 3.16, such a set is of the form $SS_M(l)$, where $l$ is a recoverable literal of $KB$. The converse direction: let $l$ be a recoverable literal of $KB$. We have shown that there is an mcm $M$ of $KB$ such that $SS_M(l)$ is a support set of $l$. By the proof of proposition 3.16 this support set is contained in some maximal support set of $l$, and so $KB$ is a recoverable. □

3.3 The damaged literals

The last class of literals according to the $\models_{con}$-categorization consists of those literals that a consistent truth value cannot be reliably attached to them (at least, not according to the most consistent models of the knowledge-base). The following theorem strengthens this intuition:

Theorem 3.18 $l$ is damaged iff there exist mcms $M_1$ and $M_2$ s.t. $M_1(l) = f$ and $M_2(l) = t$.

Proof: “If”: follows directly from the definition of damaged literals. For the converse, suppose that $p$ is the atomic part of $l$. Since $l$ is damaged iff $p$ is damaged, it suffices to prove the claim for $p$. Now, $p$ is damaged, thus there are mcms $N_1$ and $N_2$ s.t. $N_1(p) \notin \mathcal{F}$ and $N_2(\neg p) \notin \mathcal{F}$. Suppose that $M_1$ is a valuation that assigns $f$ to $p$ and is equal to $N_1$ for all the other members of $A(KB)$. Similarly, suppose that $M_2$ assigns $t$ to $p$ and is equal to $N_2$ otherwise. Since $N_1$ and $N_2$ are mcms of $KB$, so are $M_1$ and $M_2$. □

We conclude this subsection with some observations related to damaged literals:

- The existence of a support set for a damaged literal is not assured. For instance, the damaged literal $d$ of example 3.9 is a member of a support set $(S)$. Still, no support set of $KB$ supports $d$.

- Even if there are support sets for a damaged literal, there can be other subsets that support its negation: For example, in $KB = \{p, \neg p \vee q, r, \neg r \vee \neg q\}$ where $B = FOUR$, $q$ is damaged. It has a support set: $SS(q) = \{p, \neg p \vee q\}$, but there is a support set for $\neg q$ as well: $SS(\neg q) = \{r, \neg r \vee \neg q\}$.

- Consider $KB = \{p \vee q, \neg p \vee \neg q\}$. Here both $p$ and $q$ are damaged although $KB$ is a consistent set. Intuitively, this is so because there is not enough data in $KB$ about either $p$ or $q$. In particular, a literal can be damaged not just because of “over” information, but because of a lack of data as well. In both cases, however, its truth value cannot be recovered safely.

4 The minimal mcm's of $KB$

In this section we show that if one is interested only in recovering an inconsistent knowledge-base (that is, discovering the spoiled, damaged, and recoverable literals of $KB$, as well as the support sets of the latters), then it is sufficient to consider only the $\leq_{min}$-minimal models of the most consistent models (minimal mcm's, in short). The set of the minimal mcm's of $KB$ will be denoted by $\Omega(KB)$, or just $\Omega$.

Abstractly, we can view the construction of $\Omega$ as a composition of the two consequence relations $\models_{con}$ and $\models_{min}$. First, we confine ourselves to the mcm's of $KB$ by using $\models_{con}$, then we minimize the valuations that we have got by using $\models_{min}$. This process is a special case of what is called “stratification” in [BS88].
Lemma 4.1 Let KB be a finite set of normalized extended clauses. For every mcm M of KB there is an N ∈ Ω(KB) s.t. N ≤k M and IncN(KB) = IncM(KB).

Proof: Let M ∈ mcms(KB) and N ∈ Ω(KB) s.t. N ≤k M. Suppose also that IncN(KB) ≠ IncM(KB). Since M, N ∈ mcms(KB), there are q1, q2 ∈ A(KB) s.t. q1 ∈ IncN(KB) \ IncM(KB) and q2 ∈ IncM(KB) \ IncN(KB). Assume that N(q1) ∈ F. Since N(q1) ∈ I then N(¬q1) ∈ F as well. Thus M(q1) ≥κ N(q1) ∈ F and M(¬q1) ≥κ N(¬q1) ∈ F, so M(q1) ∈ I = a contradiction. Hence N(q1) /∈ F. Similarly, N(¬q1) /∈ F. Now, consider the valuation N′ that assigns t to q1, and is equal to N on every other p ∈ A(KB). It is easy to verify that for ψ ∈ KB, N′(ψ) ∈ F whenever N(ψ) ∈ F, thus N′ ∈ mod(KB). But IncN(KB) = IncN(KB) ∪ {q1}, therefore N′ >con N, and so N /∈ mcms(KB). In particular N /∈ Ω(KB) − a contradiction. □

Proposition 4.2 Let KB be a finite set of extended clauses in BL. Then:

a) i is spoiled literal in KB iff for every model M ∈ Ω(KB), M(i) ∈ F and M(¬i) /∈ F.

b) i is recoverable iff for every M ∈ Ω(KB), M(i) ∈ F, and there is N ∈ Ω(KB) s.t. N(i) ∈ F \ I.

c) i is damaged literal in KB iff there are M1, M2 ∈ Ω(KB) s.t. M1(i) /∈ F and M2(¬i) /∈ F.

Proof: We show only part (b); the proofs of the other parts are similar. Suppose that for every M ∈ Ω, M(i) ∈ F. Since F is upwards closed w.r.t. ≤k, this is true for every mcm M′, hence KB ≺con i. Since there is an N ∈ Ω s.t. N(i) ∈ F \ I, then N(¬i) /∈ F, and so KB /∈ con 7, thus i is recoverable. The other direction: since i is recoverable, it is assigned a designated truth value in every mcm of KB, in particular it is designated in every minimal mcm. Also, there must be an mcm N s.t. N(i) ∈ F \ I (otherwise i is spoiled). By lemma 4.1 there is an N′ ∈ Ω s.t. N′(i) ∈ F \ I as well. Therefore N′(i) ∈ F \ I. □

Another result is a characterization of the maximal support sets in terms of minimal mcms (cf. 3.16):

Proposition 4.3 Every maximal support set of a recoverable literal i is associated with some minimal mcm M ∈ Ω s.t. M(i) /∈ I.

Proof: Follows easily from proposition 3.16 and lemma 4.1. □

5 Extensions to first-order logic

It is possible to directly expand the present discussion to any first-order knowledge-bases provided that:

(a) there are no quantifiers within the clauses; each extended clause that contains variables is considered as universally quantified, and

(b) no function symbols are allowed. 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 member of the Herbrand universe, U.

Since we have not allowed the appearance of function symbols, and since we deal with finite knowledge-bases, U must be finite as well. Formally: KB = {ρ(ψ) | ψ ∈ KB, ρ : var(ψ) → U} where ρ is a ground substitution from the variables of every ψ ∈ KB to the individuals of Herbrand universe U.

6 Examples and discussion

In what follows we consider two benchmark problems which are given in [Li88]. The illustrations are considered in B = FOUR with I = {T}. As it is shown below, our system manages to keep the results very close to those suggested in [Li88].

Consider the following block world knowledge-base:

\[ KB1 = \{ heavy(A), heavy(B), \neg onJable(A), \neg red(B), heavy(x) \rightarrow onJable(x), heavy(x) \rightarrow red(x) \} \]

The minimal mcms of KB1 are given in figure 3.8. Their associated support sets are listed below:

\[ KB1 \text{ has 16 mcms. We omit the other 12, which are not } \leq_k \text{-minimal.} \]
The `heavy` table contains:

<table>
<thead>
<tr>
<th>mcm</th>
<th><code>heavy(A)</code></th>
<th><code>heavy(B)</code></th>
<th><code>red(A)</code></th>
<th><code>red(B)</code></th>
<th><code>on_table(A)</code></th>
<th><code>on_table(B)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>M1a</code></td>
<td><code>t</code></td>
<td><code>t</code></td>
<td><code>-</code></td>
<td><code>-</code></td>
<td><code>F</code></td>
<td><code>F</code></td>
</tr>
<tr>
<td><code>M1b</code></td>
<td><code>t</code></td>
<td><code>T</code></td>
<td><code>t</code></td>
<td><code>-</code></td>
<td><code>F</code></td>
<td><code>T</code></td>
</tr>
<tr>
<td><code>M1c</code></td>
<td><code>T</code></td>
<td><code>T</code></td>
<td><code>T</code></td>
<td><code>-</code></td>
<td><code>F</code></td>
<td><code>T</code></td>
</tr>
<tr>
<td><code>M1d</code></td>
<td><code>T</code></td>
<td><code>T</code></td>
<td><code>T</code></td>
<td><code>-</code></td>
<td><code>T</code></td>
<td><code>T</code></td>
</tr>
</tbody>
</table>

Figure 3: The minimal mcm's of KB1

- `KB1a` = \{`heavy(A)`, `heavy(B)`, `heavy(A)` → `red(A)`, `heavy(B)` → `on_table(B)`\}
- `KB1b` = \{`heavy(A)`, ¬`red(B)`, `heavy(A)` → `red(A)`\}
- `KB1c` = \{¬`on_table(A)`, `heavy(B)`, `heavy(B)` → `on_table(B)`\}
- `KB1d` = \{¬`on_table(A)`, ¬`red(B)`\} \(^9\)

`KB1a` seems to be the preferable support sets according to many criteria: It is the largest set, it supports more literals than any other support set, and it contains maximal information in the sense of \[Lo94\] \(^10\). `KB1a` implies that `on_table(B)` and `red(A)`. These are also the conclusions in \[Li88, problem A3\].

For another example of the block world, consider the following knowledge-base:
- `KB2` = \{`heavy(A)`, `heavy(B)`, `heavy(C)`, `heavy(x)` → `on_table(x)`, ¬`on_table(A)` ∨ ¬`on_table(B)`\}

Note that the last assertion in `KB2` states that there is an unknown exception in the information. The mcm's of `KB2` are given in figure 4.

<table>
<thead>
<tr>
<th>mcm</th>
<th><code>heavy(A)</code></th>
<th><code>heavy(B)</code></th>
<th><code>heavy(C)</code></th>
<th><code>on_table(A)</code></th>
<th><code>on_table(B)</code></th>
<th><code>on_table(C)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>M2a</code></td>
<td><code>T</code></td>
<td><code>t</code></td>
<td><code>t</code></td>
<td><code>f</code></td>
<td><code>t</code></td>
<td><code>t</code></td>
</tr>
<tr>
<td><code>M2b</code></td>
<td><code>t</code></td>
<td><code>-</code></td>
<td><code>-</code></td>
<td><code>-</code></td>
<td><code>-</code></td>
<td><code>-</code></td>
</tr>
<tr>
<td><code>M2c</code></td>
<td><code>t</code></td>
<td><code>T</code></td>
<td><code>t</code></td>
<td><code>f</code></td>
<td><code>f</code></td>
<td><code>t</code></td>
</tr>
<tr>
<td><code>M2d</code></td>
<td><code>t</code></td>
<td><code>t</code></td>
<td><code>T</code></td>
<td><code>t</code></td>
<td><code>t</code></td>
<td><code>t</code></td>
</tr>
</tbody>
</table>

Figure 4: The (minimal) mcm's of `KB2`

Hence, `heavy(X)` for `X = A, B, C` and `on_table(C)` are all recoverable, while `on_table(A)` and `on_table(B)` are damaged. The support sets of `KB2` are listed below:

- \{`heavy(B)`, `heavy(C)`, `heavy(B)` → `on_table(B)`, `heavy(C)` → `on_table(C)`, ¬`on_table(A)` ∨ ¬`on_table(B)`\}
- \{`heavy(A)`, `heavy(C)`, `heavy(A)` → `on_table(A)`, `heavy(C)` → `on_table(C)`, ¬`on_table(A)` ∨ ¬`on_table(B)`\}
- \{`heavy(A)`, `heavy(B)`, `heavy(C)`, `heavy(A)` → `on_table(A)`, `heavy(B)` → `on_table(B)`, `heavy(C)` → `on_table(C)`\}
- \{`heavy(A)`, `heavy(B)`, `heavy(C)`, `heavy(A)` → `on_table(A)`, `heavy(C)` → `on_table(C)`\}

Note that every recovered knowledge-base preserves the intuitive conclusions of `KB2`, i.e.: (a) block `C` is on the table, and (b) either block `A` or block `B` is on the table, but there is no evidence that both are on the table. Again, these conclusions are similar to those of \[Li88\].

Due to the lack of space we have not considered here all the benchmarks of \[Li88\]. We confined ourselves to two representative examples of category A (default reasoning). However, the reader might want to check that many other test criteria mentioned there are met in our system. Most notable are the inheritance features, and the autoepistemic characterizations.

\(^9\) The “conservative” nature of the system is emphasized here: each solution gets rid of the information it considers as contradictory, and leaves all the other data unchanged.

\(^{10}\) See \[AA94\] for a detailed discussion of methods for choosing the preferred support set.
References


[Su90a] V.S. Subrahmanian. Mechanical proof procedures for many valued lattice-based logic programming. Journal of Non-Classical Logic, Vol.7 (pp. 7-41); 1990.

[Su90b] V.S. Subrahmanian. Paraconsistent disjunctive deductive databases. IEEE 20th Int. Symp. on Multiple-Valued Logic (pp.339-345); 1990.