An Algorithmic Approach to Recover Inconsistent Knowledge-bases

Ofer Arieli

Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
arieli@cs.kuleuven.ac.be

Abstract. We consider an algorithmic approach for revising inconsistent data and restoring its consistency. This approach detects the "spoiled" part of the data (i.e., the set of assertions that cause inconsistency), deletes it from the knowledge-base, and then draws classical conclusions from the "recovered" information. The essence of this approach is its coherence with the original (possibly inconsistent) data: On one hand it is possible to draw classical conclusions from any data that is not related to the contradictory information, while on the other hand, the only inferences allowed by this approach are those that do not contradict any former conclusion. This method may therefore be used by systems that restore consistent information and are obliged to their resource of information. Common examples of this case are diagnostic procedures that analyse faulty components of malfunction devices, and database management systems that amalgamate distributed knowledge-bases.

1 Motivation

In this paper we introduce an algorithmic approach to revise inconsistent information and restore its consistency. This approach (sometimes called "coherent" [5], or "conservative" [15]) considers contradictory data as useless, and uses only a consistent part of the original information for making inferences. To see the rationality behind this approach consider, for instance, the following set of propositional assertions:

$$KB = \{p,\ \neg p,\ \neg p \vee q,\ r,\ \neg r \vee s\}.$$

Since $\neg p$ is true in KB, so is $\neg p \vee q$ (even if q is false), and so a plausible inference mechanism should not apply here the Disjunctive Syllogism to p and $\neg p \vee q$. Intuitively, this is so since the information regarding p is contradictory, and so one should not rely on it for drawing inferences. On the other hand, applying the Disjunctive Syllogism to $\{r,\ \neg r \vee s\}$ may be justified by the fact that this subset of formulae should *not* be affected by the inconsistency in KB, therefore inference rules that are classically valid can be applied to it.

The two major goals of coherent approaches in general, and our formalism in particular, are therefore the following:

a) Detect and isolate "spoiled" parts of the knowledge-base, i.e.: Remove from the knowledge-base subsets of assertions that cause inconsistency,
b) Draw classical conclusions in a non-trivial way from any data that is not related to the contradictory information. Such inferences should be semantically coherent with the original data, that is: Only inferences that do not contradict any previously drawn conclusions are allowed.

For achieving the goals above we consider an algorithmic approach that is based on a four-valued semantics [3, 4]. Using a multiple-valued semantics is a common way to overcome the shortcomings of classical calculus (see, e.g., [3, 6, 7, 12–14]), and as we shall see in what follows, four-valued semantics is particularly suitable for our purpose.

A similar algorithmic approach for recovering stratified knowledge-base, which is also based on a four-valued semantics, was introduced in [1, 2]. Here we generalize and improve that approach in the sense that we consider a better search engine, and provide and algorithm that recovers *arbitrary* knowledge-bases rather than only stratified ones.

2 Background

2.1 Belnap four-valued lattice

Our method is based on Belnap's well-known algebraic structure, introduced in [3, 4]. This structure consists of four truth values: the classical ones (t, f), a truth value $(\bot)$ that intuitively represents lack of information, and a truth value $(\top)$ that may intuitively be understood as representing contradictions. These four elements are simultaneously ordered in two distributive lattices. In one of them, denoted by $L_4 = (\{t, f, \top, \bot\}, \leq_t)$, f is the $\leq_t$-minimal element, t is the $\leq_t$-maximal one, and $\bot, \top$ are two intermediate values that are incomparable. The partial order of this lattice may be intuitively understood as representing differences in the amount of *truth* of each element. In the other lattice, denoted by $A_4 = (\{t, f, \top, \bot\}, \leq_k)$, $\bot$ is the $\leq_k$-minimal element, $\top$ is the $\leq_k$-maximal one, and t, f are two intermediate values. The partial order $\leq_k$ of this lattice intuitively represents differences in the amount of *knowledge* (or information) that each element exhibits. We denote Belnap four-valued structure together with its two partial orders by $\mathcal{FOUR}$ (see Figure 1).

As usual, we shall denote the $\leq_t$-meet and the $\leq_t$-join of $\mathcal{FOUR}$ by $\wedge$ and $\vee$, respectively. In addition, we shall denote by $\neg$ the involution operation on $\leq_t$, for which $\neg\top = \top$ and $\neg\bot = \bot$.

2.2 Knowledge-bases: syntax and semantics

The language we use here is the standard propositional one, based on the propositional constants $t, f, \top, \bot$, and the connectives $\vee, \wedge, \neg$ that correspond, respectively, to the join, meet, and the negation operations w.r.t. $\leq_t$. Atomic formulae

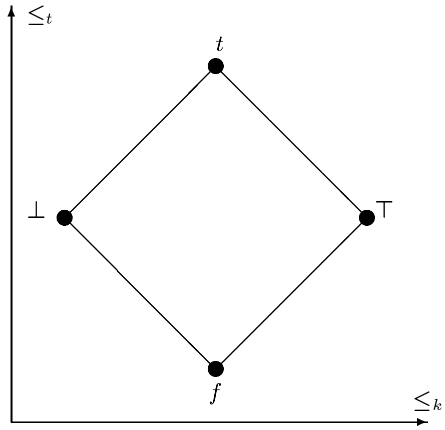

Fig. 1. Belnap lattice, $\mathcal{FOUR}$

are denoted by p, q, literals (i.e., atomic formulae or their negations) are denoted by l, and complex formulae are denoted by ψ, ϕ. Given a set S of formulae, we shall write $\mathcal{A}(S)$ to denote the set of the atomic formulae that occur in S, and $\mathcal{L}(S)$ to denote the set of the literals that occur in S ($\mathcal{A}$ and $\mathcal{L}$ denote, respectively, the set of atomic formulae and the set of literals in the language). The complement of a literal l is denoted by $\bar{l}$. An atomic formula $p \in \mathcal{A}(S)$ is called a *positive (negative) fact* of S if $p \in S$ ($\neg p \in S$). The set of all the (positive and negative) facts in S is denoted by $Facts(S)$.

The various semantic notions are defined on $\mathcal{FOUR}$ as natural generalizations of similar classical ones: A *valuation* ν is a function that assigns a truth value in $\mathcal{FOUR}$ to each atomic formula. Any valuation is extended to complex formulae in the obvious way. The set of the four-valued valuations is denoted by $\mathcal{V}$. A valuation ν *satisfies* ψ iff $\nu(\psi) \in \{t, \top\}$. t and $\top$ are called the *designated* elements of $\mathcal{FOUR}$. A valuation that satisfies every formula in a given set S of formulae is a *model* of S. A model of S will usually be denoted by M or N. The set of all the models of S is denoted by $mod(S)$.

The formulae that will be considered here are clauses, i.e.: disjunctions of literals. The following useful property of clauses is easily shown by an induction on the structure of clauses:

Lemma 1. Let ψ be a clause and ν a valuation. Then $\nu(\psi) \in \{t, \top\}$ iff there is some $l \in \mathcal{L}(\psi)$ s.t. $\nu(l) \in \{t, \top\}$.

A finite set of clauses is called a *knowledge-base*, and is denoted by KB. As the following lemma shows, representing formulae in a clause form does not reduce the generality.

Lemma 2. [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 $\phi \in S$.

Given a certain knowledge-base KB, we consider the $\leq_k$-minimal elements in $mod(KB)$. These models reflect the intuition that one should not assume what is not really represented in KB.

Definition 1. Let $\nu_1, \nu_2 \in \mathcal{V}$.

a) ν_1 is k-*smaller* than ν_2 iff for every atom p, $\nu_1(p) \leq_k \nu_2(p)$.
b) $\nu \in mod(KB)$ is a k-*minimal model* of KB if there is no other model of KB that is k-smaller than ν.

Example 1. Consider the following knowledge-base:

$$\mathcal{KB} = \{p,\ \neg q,\ \neg p \vee q,\ \neg p \vee h,\ q \vee r \vee s,\ q \vee \neg r \vee \neg s,\ h \vee r,\ h \vee s\}$$

The (k-minimal) models of $\mathcal{KB}$ are given in Table 1 below. We shall use $\mathcal{KB}$ for the demonstrations in the sequel.

Table 1. The (k-minimal) models of $\mathcal{KB}$

Model No.	p	q	h	r	s	k-minimal
M_1	t	$\top$	t	$\bot$	$\bot$	$+$
$M_2 - M_4$	t	$\top$	t	$\bot$	$f, t, \top$	
$M_5 - M_{16}$	t	$\top$	t	$f, t, \top$	$\bot, f, t, \top$	
$M_{17} - M_{32}$	t	$\top$	$\top$	$\bot, f, t, \top$	$\bot, f, t, \top$	
M_{33}	$\top$	f	$\bot$	t	$\top$	$+$
M_{34}	$\top$	f	$\bot$	$\top$	t	$+$
M_{35}	$\top$	f	$\bot$	$\top$	$\top$	
M_{36}	$\top$	f	f	t	$\top$	
$M_{37} - M_{38}$	$\top$	f	f	$\top$	$t, \top$	
M_{39}	$\top$	f	t	$\bot$	$\top$	$+$
M_{40}	$\top$	f	t	f	t	$+$
M_{41}	$\top$	f	t	f	$\top$	
M_{42}	$\top$	f	t	t	f	$+$
M_{43}	$\top$	f	t	t	$\top$	
M_{44}	$\top$	f	t	$\top$	$\bot$	$+$
$M_{45} - M_{47}$	$\top$	f	t	$\top$	$f, t, \top$	
M_{48}	$\top$	f	$\top$	$\bot$	$\top$	
$M_{49} - M_{50}$	$\top$	f	$\top$	f	$t, \top$	
$M_{51} - M_{52}$	$\top$	f	$\top$	t	$f, \top$	
$M_{53} - M_{56}$	$\top$	f	$\top$	$\top$	$\bot, f, t, \top$	
M_{57}	$\top$	$\top$	$\bot$	t	t	$+$
M_{58}	$\top$	$\top$	$\bot$	t	$\top$	
$M_{59} - M_{60}$	$\top$	$\top$	$\bot$	$\top$	$t, \top$	
$M_{61} - M_{64}$	$\top$	$\top$	f	$t, \top$	$t, \top$	
$M_{65} - M_{80}$	$\top$	$\top$	t	$\bot, f, t, \top$	$\bot, f, t, \top$	
$M_{81} - M_{96}$	$\top$	$\top$	$\top$	$\bot, f, t, \top$	$\bot, f, t, \top$	

The k-minimal models of KB will have an important role in the recovery process of KB. This may be justified by the fact that as long as one keeps the amount of information as minimal as possible, the tendency of getting into conflicts decreases.

2.3 Recovered knowledge-bases

Definition 2. Let $\nu \in \mathcal{V}$. Denote: $I(\nu) = \{p \in \mathcal{A} \mid \nu(p) = \top\}$. Usually we shall be interested in the assignments of ν w.r.t. a specific knowledge-base. In such cases we shall consider the following set: $I(\nu, KB) = \{p \in \mathcal{A}(KB) \mid \nu(p) = \top\}$.

As we have noted above, by "recovering a knowledge-base" we mean to turn it (in a plausible way) to a consistent one. That is:

Definition 3. A valuation ν is *consistent* if $I(\nu) = \emptyset$. A knowledge-base is consistent if it has a consistent model.

Proposition 1. [1, 2] A knowledge-base is consistent iff it is classically consistent.

The recovery process is based on the following notion:

Definition 4. A *recovered knowledge-base* KB' of a knowledge-base KB is a subset of KB with a consistent model M' s.t. there is a (not necessarily consistent) model M of KB, for which $M'(p) = M(p)$ for every $p \in \mathcal{A}(KB')$.

Example 2. The set $\{p\}$ is a recovered knowledge-base of $KB_1 = \{p, q, \neg q\}$, but it is not a recovered knowledge-base of $KB_2 = \{p, \neg p\}$. This example demonstrates the fact that in order to recover a given inconsistent knowledge-base, it is *not sufficient* to find some of its (maximal) consistent subset(s), but it is necessary to ensure that the subset under consideration would semantically correspond to the original, inconsistent data; In our case, $\{p\}$ does not recover KB_2 even though it is a classically consistent subset of KB_2, just because of the fact that this set contradicts an information ($\neg p$) that is explicitly stated in the original knowledge-base. Therefore, the "semantical correspondence" property is not preserved in this case.[1]

Given an inconsistent knowledge-base KB, the idea is to choose one of its recovered knowledge-bases and to treat this set as the relevant knowledge-base for deducing classical inferences. Next we show that the set of recovered knowledge-bases of KB may be easily constructed from the set of its models:

Definition 5. Let $\nu \in \mathcal{V}$. The set that is *associated with* ν is defined as follows:

$$KB_\nu = \{\psi \in KB \mid \nu(\psi) = t \text{ and } \mathcal{A}(\psi) \cap I(\nu, KB) = \emptyset\}.$$

[1] Keeping this "semantical correspondence" to the original information is one of the main differences between the present formalism and some other formalisms for restoring consistency (see, e.g., [5, 6, 9]).

The set KB_ν corresponds to the (maximal) fragment of KB that can be interpreted in a consistent way by ν. Elimination of pieces of "inadequate" information in order to get a more "robust" representation of the "intended" knowledge is a common method in belief revision and argumentative reasoning (see, e.g., [5, 6, 9]).

Proposition 2. [1] Every set that is associated with a model of KB is a recovered knowledge-base of KB.

Proposition 2 implies that usually there will be a lot of ways to recover a given inconsistent knowledge-base. By what we have noted above, plausible candidates of being the "best" recovered knowledge-base of KB would be those sets that are associated with some k-minimal model of KB.[2]

Definition 6. A set $S \subseteq KB$ is a *preferred* recovered knowledge-base of KB if it is a maximal set that is associated with some k-minimal model of KB.

Example 3. Consider again the knowledge-base $\mathcal{KB}$ of Example 1. In the notations of Table 1, the subsets of $\mathcal{KB}$ that are associated with its k-minimal models are the following:

$$\begin{aligned}
\mathcal{KB}_{M_1} &= \{p,\ \neg p \vee h,\ h \vee r,\ h \vee s\},\\
\mathcal{KB}_{M_{33}} &= \{\neg q,\ h \vee r\},\\
\mathcal{KB}_{M_{34}} &= \{\neg q,\ h \vee s\},\\
\mathcal{KB}_{M_{39}} &= \{\neg q,\ h \vee r\},\\
\mathcal{KB}_{M_{40}} &= \{\neg q,\ q \vee r \vee s,\ q \vee \neg r \vee \neg s,\ h \vee r,\ h \vee s\},\\
\mathcal{KB}_{M_{42}} &= \{\neg q,\ q \vee r \vee s,\ q \vee \neg r \vee \neg s,\ h \vee r,\ h \vee s\},\\
\mathcal{KB}_{M_{44}} &= \{\neg q,\ h \vee s\},\\
\mathcal{KB}_{M_{57}} &= \{h \vee r,\ h \vee s\}.
\end{aligned}$$

Thus, the preferred recovered knowledge-bases are $\mathcal{KB}_{M_1}$ and $\mathcal{KB}_{M_{40}} = \mathcal{KB}_{M_{42}}$.

3 Recovery of inconsistent knowledge-bases

In this section we introduce an algorithm for recovering inconsistent knowledge-bases, and consider some of its properties.

Definition 7. Let KB be a knowledge-base, and let ν be a four-valued partial valuation defined on (a subset of) $\mathcal{A}(KB)$. The *dilution* of KB w.r.t. ν (notation: $KB \downarrow \nu$) is constructed from KB by the following transformations:

1. Deleting every $\psi \in KB$ that contains either t, $\top$, or a literal l s.t. $\nu(l) \in \{t, \top\}$,
2. Removing from every formula that remains in KB every occurrence of f, $\bot$, and every occurrence of a literal l such that $\nu(l) \in \{f, \bot\}$.

[2] See [2] for some other preference criteria for choosing recovered knowledge-base.

The intuition behind the dilution process resembles, in a way, that of the Gelfond–Lifschitz transformation [8]: Any data that has no effect on the rest of the process is eliminated. Thus, for instance, if a literal l in a formula ψ is assigned a designated value, then Lemma 1 assures that eventually ψ would also have a designated value, no matter what would be the values of the elements in $\mathcal{L}(\psi) \setminus \{l\}$. Hence, these elements can be disregarded in the rest of the construction, as indeed indicated by item (1) of Definition 7. The rationality behind item (2) of the same definition is similar.

Figure 2 contains a pseudo-code of the recovery algorithm. [3] [4] As we show in Theorems 1 and 2 below, given a certain knowledge-base KB as an input, the algorithm provides the valuations needed for constructing the preferred recovered knowledge-bases of KB.

It is easy to verify that the algorithm indeed halts for every knowledge-base. This is so since knowledge-bases are finite, and since for every set S of clauses and every partial valuation ν on $\mathcal{A}(S)$, we have that $\mathcal{A}(S \downarrow \nu) \subset \mathcal{A}(S)$.

Example 4. Figure 3 below demonstrates the execution of the algorithm on the knowledge-base $\mathcal{KB}$ of the canonical example (1 and 3). In this figure we denote by $p\!:\!x$ the fact that an atom p is assigned a value x.

In the notations of Table 1, the two leftmost paths in the tree of Figure 3 produce the k-minimal model M_1, and the other paths produce the k-minimal models M_{40} and M_{42}.[5] As noted in Example 3, these are exactly the models with whom the preferred recovered knowledge-bases of $\mathcal{KB}$ are associated. By Theorem 2, these are *all* the preferred recovered knowledge-bases of $\mathcal{KB}$.

Proposition 3. Let ν be a four-valued valuation produced by the algorithm of Figure 2 for a given knowledge-base KB. Then ν is a model of KB.

Proof: Let $\psi \in KB$. By Definition 7 and the specifications of the algorithm in Figure 2, it is obvious that at some stage of the algorithm ψ is eliminated from the set of clauses as a result of a dilution on this set. Note that a formula cannot be eliminated by successively removing every literal of it according to condition (2) of Definition 7, since the last literal that remains must be assigned a designated value. Thus there must be some $l \in \mathcal{L}(\psi)$ that is assigned a designated value. By Lemma 1, then, $\nu(\psi) \in \{t, \top\}$, and so $\nu \in mod(KB)$. □

[3] The first parameter of the first call to `Recover` is the dilution of KB w.r.t. the empty valuation. This is so in order to take care of the propositional constants that appear in KB (for instance, if $p \vee f \in KB$ then $p \in KB \downarrow \emptyset$).

[4] If the knowledge-base under consideration contains clauses that are logically equivalent to f or $\bot$ (e.g., $f \vee \bot$), then in $KB \downarrow \emptyset$ such clauses will become empty. One can easily handle such degenerated cases by adding to the algorithm a line that terminates its execution once an empty clause is detected.

[5] Later on we shall take care of the redundancy.

```
input: A knowledge-base KB.
Mods = Recover(KB$\downarrow\emptyset$, $\emptyset$);
do ($\forall$M $\in$ Mods) {
   KB$_M$ = {$\psi\in$ KB | $\neg\exists$p $\in\mathcal{A}$(KB) such that M(p) = $\top$};
   output(KB$_M$);
}

procedure Recover(S,$\nu$)
/* S = a finite set of clauses, $\nu$ = the valuation constructed so far */
{
   if (S == $\emptyset$) then return($\nu$)        /* $\nu$ is a $k$-minimal model of KB */
   pos = {p $\in\mathcal{A}$(S) | p $\in$ S };        /* the positive facts in S */
   neg = {p $\in\mathcal{A}$(S) | $\neg$p $\in$ S };       /* the negative facts in S */
   if (pos $\cup$ neg == $\emptyset$) {
       do ($\forall$p $\in\mathcal{A}$(S)) {
           pick p;
           if (p $\in\mathcal{L}$(S)) then Recover(S $\cup$ {p}, $\nu$);
           if ($\neg$p $\in\mathcal{L}$(S)) then Recover(S $\cup$ {$\neg$p}, $\nu$);
       }
   }
   do ($\forall$p $\in$ (pos $\cap$ neg)) {
       pick p;
       $\mu$(p) = $\top$;
       S$'$ = S $\downarrow\mu$;
       do ($\forall$q $\neq$ p such that q $\in\mathcal{A}$(S) $\setminus$ $\mathcal{A}$(S$'$))
           $\mu$(q) = $\bot$;
       Recover(S$'$, $\nu\cup\mu$);
   }
   do ($\forall$p $\in$ (pos $\cup$ neg) $\setminus$ (pos $\cap$ neg)) {
       pick p;
       if (p $\in$ pos) then $\mu$(p) = $t$ else $\mu$(p) = $f$;
       S$'$ = S $\downarrow\mu$;
       do ($\forall$q $\neq$ p such that q $\in\mathcal{A}$(S) $\setminus$ $\mathcal{A}$(S$'$))
           $\mu$(q) = $\bot$;
       Recover(S$'$, $\nu\cup\mu$);
   }
}
```

Fig. 2. An algorithm for recovering knowledge-bases

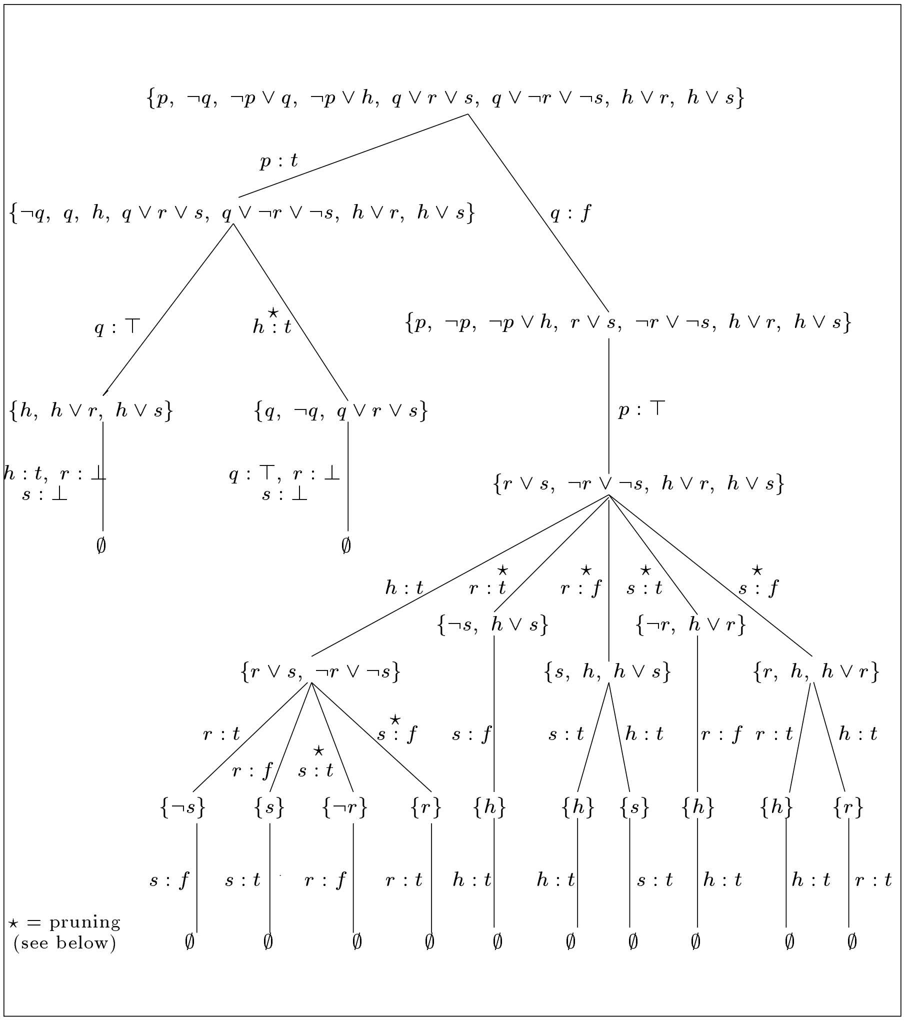

Fig. 3. Execution of the algorithm w.r.t. the canonical example

The next proposition indicates that the valuations produced by the algorithm of Figure 2 assign designated truth values only to a minimal amount of literals (no more literals than what is really necessary for providing a model for KB). In a sense, this means that a minimal amount of knowledge (or belief) is assumed.

Proposition 4. Let ν be a four-valued valuation produced by the algorithm of Figure 2 for a given knowledge-base KB. Then ν is a *choice function* on KB: For every $\psi \in KB$ there is exactly *one* literal $l \in \mathcal{L}(\psi)$ s.t. $\nu(l)$ is designated.

Proof: The proof is by an easy inspection on the execution of the algorithm. Consider some $\psi \in KB$. Suppose that it is eliminated at the i-th inductive call to `Recover`. Then all the literals $l \in \mathcal{L}(\psi)$ for which $\nu(l)$ is defined until the i-th recursive call to `Recover` has the property that $\nu(l) = f$ (otherwise ψ would have already been eliminated). Then there is some $l \in \mathcal{L}(\psi)$ (which is chosen during the i-th execution of `Recover`), for which $\nu(l) \in \{t, \top\}$, and after the next dilution ψ is eliminated, i.e.: all the rest of the literals in $\mathcal{L}(\psi)$ are assigned $\bot$. It follows, then, that every clause has a unique literal that is assigned a designated value by ν. □

Here is another evidence to the fact that only a minimal knowledge is assumed by the valuations produced by our algorithm:

Theorem 1. Let ν be a four-valued valuation produced by the algorithm of Figure 2 for a given knowledge-base KB. Then ν is a k-minimal model of KB.

Proof: First, by Proposition 3, ν is a model of KB. It remains to show, then, that ν is a k-*minimal* among the models of KB. For that consider the following set of knowledge-bases:

$$KB_0 = KB \downarrow \emptyset, \qquad KB_{i+1} = KB_i \downarrow \nu_i$$

where ν_i $(i \geq 0)$ is the partial valuation determined during the i-th recursive call to `Recover`.[6] Now, let us first assume that there is at least one (positive or negative) fact in KB (i.e., there is a literal $l \in \mathcal{L}(KB)$ s.t. $l \in KB$). We show that ν is a k-minimal model of KB by an induction on the number n of the recursive calls to `Recover` that are required for creating ν.

- $n = 0$: ν_0 may assign $\top$ only to a literal l s.t. $l \in KB$ and $\overline{l} \in KB$, while all the other elements in $\mathcal{A}(KB)$ are assigned $\bot$. In this case $\top$ is the only possible value for l, and so ν is k-minimal. The same argument is true for any literal l s.t. $l \in KB$ and $\overline{l} \notin KB$ (for that l, $\nu(l) = t$). It is also obviously true for all 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 $\mathcal{A}(KB_1)$, and suppose first that M_1 is a model of KB_1. By the induction hypothesis ν_1 is a k-minimal model of KB_1, thus there exists $p \in \mathcal{A}(KB_1)$, 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 a model of KB_1. In this case there must be a clause $\psi_1 \in KB_1$

[6] Thus, if the algorithm terminates after n recursive calls to `Recover`, then $\nu = \bigcup_{i=1}^{n} \nu_i$.

s.t. $M_1(\psi_1) \notin \{t, \top\}$. Since M is a model of KB, then by Lemma 1 there is a $\psi \in KB$ 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 KB and so $\psi_1 \notin KB_1$), while $M(l) \in \{t, \top\}$. It follows that $M(l) \not<_k \nu(l)$, therefore $M \not<_k \nu$ in this case also.

To conclude, it remains to handle the case where there are no facts in KB. In this case our algorithm operates on $KB' = KB \cup \{l\}$ for some $l \in \mathcal{L}(KB)$. But now there *is* a fact in KB', and so by what we have shown above our algorithm produces a k-minimal model for KB'. Denote this model by ν'. We have to show that ν' is also a k-minimal model of KB. Indeed, ν' is clearly a model of KB. Let M be some other model of KB. If $M(l) \in \{t, \top\}$ then M is a model of KB' and so $M \not<_k \nu'$. Otherwise, $M(l) \in \{f, \bot\}$. Consider the subset of formulae of KB in which l appears as a literal: $KB(l) = \{\psi \in KB \mid l \in \mathcal{L}(\psi)\}$. Since $l \in \mathcal{L}(KB)$, it follows that $KB(l) \neq \emptyset$. Moreover, since we assume that there are no facts in KB, in particular $l \notin KB$ and $\bar{l} \notin KB$, thus $KB(l) \not\subseteq \{l, \bar{l}\}$. Now, by the definition of ν' as a valuation that is produced by our algorithm, for every $p \in \mathcal{A}(KB(l))$ s.t. $p \neq l$, we have that $\nu'(p) = \bot$. (Such p exist since $KB(l) \neq \emptyset$ and $KB(l) \not\subseteq \{l, \bar{l}\}$. These atoms are assigned $\bot$ since all the formulae in $KB(l)$ are removed after the first dilution of KB'). Now, since we assumed that $M(l) \in \{f, \bot\}$, then by Lemma 1 there must exist some $p_0 \in \mathcal{A}(KB(l))$ s.t. $M(p_0) \in \{t, \top\}$ (Otherwise $\forall \psi \in KB(l)\ M(\psi) \notin \{t, \top\}$ and so M cannot be a model of KB). Thus $M(p_0) >_k \bot = \nu'(p_0)$ and once again we have that $M \not<_k \nu'$. □

Using Theorem 1 we can now show that the algorithm indeed properly recovers inconsistent knowledge-bases.

Theorem 2. For a given knowledge-base KB, the algorithm of Figure 2 produces *all* the valuations ν, for which KB_ν is a preferred recovered knowledge-base of KB.

Proof: By Theorem 1, if ν is obtained by our algorithm, then KB_ν is an element of the following set:

$$\Omega = \{KB_M \mid M \text{ is a } k\text{-minimal model of } KB\}.$$

It remains to show, therefore, that the algorithm produces valuations ν_j, for which KB_{ν_j} are the *maximal* elements of Ω. Indeed, given a k-minimal model M of KB, we show that the algorithm produces a valuation ν s.t. $I(\nu, KB) \subseteq I(M, KB)$, and therefore $KB_M \subseteq KB_\nu$.

As in Theorem 1, we denote by ν_i the partial valuation that is determined during stage i of the algorithm (thus, if the algorithm terminates after n stages, then $\nu = \cup_{i=1}^{n} \nu_i$), and M_i is the reduction of M to the literals on which ν_i is defined. Also, we use the following notations: $KB_0 = KB \downarrow \emptyset$, and for every $i \geq 0$, $KB_{i+1} = KB_i \downarrow \nu_i$. Now, suppose first that $\mathit{Facts}(KB_0) \neq \emptyset$ (i.e., there is some [positive or negative] fact in KB_0). If $\{l, \bar{l}\} \subseteq \mathit{Facts}(KB_0)$ for some literal l, set $\nu_0(l) = \top$ (note that in this case necessarily $M(l) = \top$ as well, since M is a model of KB and so it must assign $\top$ to all the facts of KB that are both positive and negative). Otherwise, choose some $l \in Facts(KB_0)$ s.t. $M(l)=t$ (such a literal must exist, since M is a model of KB and so it must assign designated values to the facts of KB_0), and set $\nu_0(l)=t$. If $Facts(KB_0)$ is empty, then if there is some $l \in \mathcal{L}(KB_0)$ s.t. $M(l)=t$ set $\nu_0(l)=t$ as well. Otherwise, pick some $l \in \mathcal{L}(KB_0)$ s.t. $M(l)=\bot$ and set $\nu_0(l)=t$ (there must be such a literal, since otherwise $\forall l \in \mathcal{L}(KB_0)\ M(l) \in \{\top, f\}$ and since $Facts(KB_0)=\emptyset$, this implies that M is not k-minimal, since one can easily construct a model of KB which is k-smaller than M by changing one of the f-assignments of M to $\bot$, or one of the $\top$-assignments of M to t). Now, in order to determine ν_1 we follow a similar procedure, this time for KB_1: If $Facts(KB_1) \neq \emptyset$ then if $\{l, \overline{l}\} \subseteq Facts(KB_1)$ for some l, set $\nu_1(l)=\top$ (note that in this case necessarily $M(l)=\top$ as well, since by the construction of ν_0, we have that $KB_1 = KB \downarrow \nu_0 \subseteq KB \downarrow M_0$, and so $\{\overline{l}, l\} \subseteq KB \downarrow M_0$ as well, which means that M must assign l the value $\top$ in order to be a model of KB). Otherwise, if there is some $l \in Facts(KB_1)$ s.t. $M(l)=t$ set $\nu_1(l)=t$ as well. Otherwise, pick some $l \in Facts(KB_1)$ s.t. $M(l) \in \bot$ (again, such an l must exists. Otherwise, by the same reasons considered above, we will have a contradiction to the fact that M is a k-minimal model of KB), and set $\nu_1(l)=t$. The procedure in case that $Facts(KB_1)=\emptyset$ is the same as the one in case that $Facts(KB_0)=\emptyset$.

Now, repeat the same process until for some n, KB_n becomes empty. Let $\nu = \cup_{i=1}^{n} \nu_i$. The following two facts are easily verified:

1. In the process of creating ν we followed the execution of the algorithm along one path of its search tree. Hence ν is obtained by our algorithm when KB is given as its input.
2. If $\nu(l)=\top$ then $M(l)=\top$ as well (see the notes whenever $\nu_i(l)=\top$).

By (2), $I(\nu, KB) \subseteq I(M, KB)$, and so $KB_M \subseteq KB_\nu$. Thus, by (1), an output ν of the algorithm corresponds to a preferred recovered knowledge-base KB_ν of KB. □

Clearly, large knowledge-bases that contain a lot of contradictory information may be recovered in many different ways. Therefore, computing all the preferred recovered knowledge-bases in such cases might require a considerable amount of running time. It is worth noting, however, that *arbitrary recovery* of a given knowledge-base KB (i.e., producing *some* preferred recovered knowledge-base of KB) obtains quite easily. This is so since the execution time for producing the first output (valuation) is bounded by $O(|\mathcal{L}(KB)| \cdot |KB|)$; A construction of the first output requires no more than $|\mathcal{L}(KB)|$ calls to `Recover` (as there are no more than $|\mathcal{L}(KB)|$ picked literals), and each call takes no more than $O(|KB|)$ running time.

We conclude this section with some notes on practical ways to reduce the execution time of the algorithm.

A. Pruning of the search tree

Let us consider once again the search tree of Figure 3. Denote the paths in this tree from the leftmost righthand by $1, \ldots, 12$. Clearly, paths 1 and 2 yield the same result. Similarly, the same valuation is produced in paths 3,6,7,11,12, and the remaining paths in the search tree also yield the same valuation. It is possible to avoid such duplications by performing a backtracking once we find out that we are constructing a valuation which is the same as another valuation that has already been produced before. Indeed, note that a path i in the search three of the algorithm corresponds to a sequence of partial valuations $\nu_0^i, \nu_1^i, \ldots, \nu_{n_i}^i$ that are constructed along its nodes. Thus, if we denote by $\mathcal{A}(KB)[\mu]$ the elements of $\mathcal{A}(KB)$ on which the partial valuation μ is defined, then it is possible to terminate the j-th flow of the algorithm (terminology: to *prune* the j-th subtree) at stage m iff there is a flow $i<j$, s.t. $\bigcup_{k=1}^{m} \mathcal{A}(KB)[\nu_k^i] = \bigcup_{k=1}^{m} \mathcal{A}(KB)[\nu_k^j]$.

Example 5. In Figure 3 the pruning locations (in paths 2, 5–12) are marked with an asterisk. Thus, only paths 1, 3, and 4 of the search tree are not pruned. They yield, respectively, the k-minimal models M_1, M_{42}, and M_{40} of $\mathcal{KB}$.[7]

Obviously, the pruning consideration might drastically improve the search mechanism of the algorithm. The tradeoff is that for checking the pruning condition we have to use much more memory space, since the algorithm has to keep tracks to valuations that correspond to previous search flows.

B. Handling unrelated information

There are many cases in which a new information should not affect any previous conclusion.[8] In such cases a plausible mechanism of belief revision should not retract any previous conclusion. Therefore, the general expectation is that in these cases the computational complexity of adding the new data to the knowledge-base and computing its new consequences would be relatively low. Detecting those cases and finding an appropriate methodology to handle them is sometime called “the irrelevance problem”. In the next proposition we show that in cases where a totally irrelevant information arrives, it is possible to avoid executing the recovery algorithm; The new data can safely be added to any preferred recovered knowledge-base without damaging any of its properties.

Proposition 5. Let KB_1 and KB_2 be two subsets of a knowledge-base KB that satisfy the following conditions:

(a) $KB_1 \cup KB_2 = KB$, (b) $\mathcal{A}(KB_1) \cap \mathcal{A}(KB_2) = \emptyset$,[9] (c) KB_1 is consistent.

[7] As noted in Example 3, these are exactly the models with whom the prefered recovered knowledge-bases of $\mathcal{KB}$ are associated.

[8] This is the case, for instance, where there is no evidence of any relation between the new data and the old one.

[9] In case that conditions (a) and (b) are satisfied we say that KB_1 and KB_2 are a *partition* of KB.

If S is a preferred recovered knowledge-base of KB_2, then $S \cup KB_1$ is a preferred recovered knowledge-base of KB.

Proof: For the proof we need the following result:

Lemma 5-A: [1, 2] For every model M of a knowledge-base KB there is a k-minimal model M' of KB s.t. $M' \leq_k M$.[10]

Suppose now that S is a preferred recovered knowledge-base of KB_2. Then it is associated with some k-minimal model ν_2 of KB_2, i.e. $S = (KB_2)_{\nu_2}$. Also, since KB_1 is classically consistent, it has a classical model, denote it ν_1. Now, consider a valuation ν that is defined for every atomic formula p as follows:

$$\nu(p) = \begin{cases} \nu_1(p) & \text{if } p \in \mathcal{A}(KB_1) \\ \nu_2(p) & \text{if } p \in \mathcal{A}(KB_2) \end{cases}$$

Since $\mathcal{A}(KB_1) \cap \mathcal{A}(KB_2) = \emptyset$, ν is well defined. It is also easy to see that ν is a model of KB, and that $KB_\nu = (KB_2)_{\nu_2} \cup (KB_1)_{\nu_1} = S \cup KB_1$. By Lemma 5-A there is a k-minimal model M of KB s.t. $M \leq_k \nu$. In particular, $I(M, KB) \subseteq I(\nu, KB)$, and so $KB_\nu \subseteq KB_M$. But $KB_\nu = S \cup KB_1$, and since S is a maximal recovered knowledge-base of KB_2, KB_ν must be a maximal recovered knowledge-base of KB. Thus $KB_M = KB_\nu = S \cup KB_1$ is a maximal recovered knowledge-base of KB and it is associated with a k-minimal model of KB. Hence $S \cup KB_1$ is indeed a preferred recovered knowledge-base of KB. □

Note that an immediate consequence of Proposition 5 is that in case that KB is classically consistent, then KB itself is the (only) preferred recovered knowledge-base, as indeed one expects.

Example 6. Consider again our canonical example (1, 3, 4). Let $\mathcal{KB}' = \mathcal{KB} \cup \{u, \neg v \vee w\}$. The prefered recovered knowledge-bases of $\mathcal{KB}'$ are simply obtained by adding $\{u, \neg v \vee w\}$ to each prefered recovered knowledge-base of $\mathcal{KB}$. I.e., the preferred recovered knowledge-bases of $\mathcal{KB}'$ are $\{p,\ \neg p \vee h,\ h \vee r,\ h \vee s,\ u,\ \neg v \vee w\}$ and $\{\neg q,\ q \vee r \vee s,\ q \vee \neg r \vee \neg s,\ h \vee r,\ h \vee s,\ u,\ \neg v \vee w\}$.

It follows that in many cases it is possible to drastically reduce the execution time of the algorithm: If the knowledge-base under consideration can be partitioned into two subsets such that one of them is classically consistent, then in order to recover the knowledge-base it is sufficient to activate the algorithm only on the inconsistent subset, and then to add the consistent set to every preferred recovered knowledge-base that is obtained by the algorithm.

4 Conclusion

In this work we have introduced a simple algorithmic method for restoring the consistency of inconsistent knowledge-bases. Restoration of consistent data is a

[10] This property is sometimes called *smoothness* [10] or *stopperdness* [11].

key concept in many applications, such as model-base diagnostic systems, database management systems for distributed (and possibly contradicting) sources of information, and pre-processing phases of procedures for a (classical) automated deduction. In all these areas, then, the techniques discusses in this paper may be useful.

We have addressed here the propositional case in which our algorithm can easily be implemented in practice. Its computational complexity in the general case, and further practical considerations for an efficient handling of first-order languages, remain to be studied.

Acknowledgement

I would like to thank the anonymous referees for their helpful comments. This work is supported by the Visiting Postdoctoral Fellowship FWO Flanders.

References

1. O.Arieli, A.Avron. *Four-valued diagnoses for stratified knowledge-bases.* Proc. CSL'96, Selected Papers (D.Van-Dalen, M.Bezem, editors), Springer Verlag, LNCS No.1258, pages 1–17, 1997.
2. O.Arieli, A.Avron. *A model theoretic approach to recover consistent data from inconsistent knowledge-bases.* Journal of Automated Reasoning 22(3), pages 263–309, 1999.
3. N.D.Belnap. *A useful four-valued logic.* Modern Uses of Multiple-Valued Logic (G.Epstein, J.M.Dunn, editors), Reidel Publishing Company, pages 7–37, 1977.
4. N.D.Belnap. *How computer should think.* Contemporary Aspects of Philosophy (G.Ryle, editor), Oriel Press, pages 30–56, 1977.
5. S.Benferhat, D.Dubois, H.Prade. *How to infer from inconsistent beliefs without revising?* Proc. IJCAI'95, pages 1449–1455, 1995.
6. D.Dubois, J.Lang, H.Prade. *Possibilistic logic.* Handbook of Logic in Artificial Intelligence and Logic Programming (D.Gabbay, C.Hogger, J.Robinson, editors), Oxford Science Publications, pages 439–513, 1994.
7. M.Fitting. *Kleene's three-valued logics and their children.* Fundamenta Informaticae 20, pages 113–131, 1994.
8. M.Gelfond, V.Lifschitz. *The stable model semantics for logic programming.* Proc. of the 5th Logic Programming Symp. (R.Kowalski, K.Browen, editors) MIT Press, Cambridge, MA, pages 1070–1080, 1988.
9. M.Ginsberg. *Counterfactuals.* Artificial Intelligence 30(1), pages 35–79, 1986.
10. S.Kraus, D.Lehmann, M.Magidor. *Nonmonotonic reasoning, preferential models and cumulative logics.* Artificial Intelligence 44(1–2), pages 167–207, 1990.
11. D.Makinson. *General patterns in nonmonotonic reasoning.* Handbook of Logic in Artificial Intelligence and Logic Programming 3 (D.Gabbay, C.Hogger, J.Robinson, editors), Oxford Science Publishers, pages 35–110, 1994.
12. J.Pearl. *Reasoning under uncertainty.* Annual Review of Computer Science 4, pages 37–72, 1989.
13. G.Priest. *Minimally Inconsistent LP.* Studia Logica 50, pages 321–331, 1991.
14. V.S.Subrahmanian. *Mechanical proof procedures for many valued lattice-based logic programming.* Journal of Non-Classical Logic 7, pages 7–41, 1990.
15. G.Wagner. *Vivid logic.* Springer-Verlag, LNAI No.764, 1994.