View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Open Archive Toulouse Archive Ouverte

Answering queries adressed to several databases according to a majority merging approach

Laurence Cholvy and Christophe Garion

ONERA-Toulouse
2 bis avenue Edouard Belin
BP 4025
31055 Toulouse Cedex 4
France
{ cholvy, garion }@cert.fr

Abstract

The general context of this work is the problem of merging data provided by several sources which can be contradictory. Focusing on the case when the information sources do not contain any disjunction, this paper first defines a propositional modal logic for reasoning with data obtained by merging several information sources according to a majority approach. Then it defines a theorem prover to automatically deduce these merged data. Finally, it shows how to use this prover to implement a query evaluator which answers queries adressed to several databases. This evaluator is such that the answer to a query is the one that could be computed by a classical evaluator if the query was adressed to the merged databases. The databases we consider are made of an extensional part, i.e. a set of positive or negative ground literals, and an intensional part i.e. a set of first order function-free clauses. A restriction is imposed to these databases in order to avoid disjunctive data.

Keywords: Database merging, majority merging, logic, deductive databases.

1 Introduction

The problem of merging information sources has been intensively studied for some years [BKMS91], [BKMS92], [Cho93], [Sub94], [Lin96], [Cho98a], [LM98], [SDL+98], [KPP98], [KPP99], [Lia00]. This is due to the growing number of applications in which accessing several information sources to make a decision is needed. The main problem in dealing with multiple information sources is the possible inconsistency between sources.

The many works which address this problem show that there is not an unique method for merging information.

Obviously, the adequate merging process depends on the type of the information to be merged. This information can be beliefs the sources have about the real world and in this case, the aim of the merging process is to refine our perception of the real world. But this information can also be a description of a world that is considered to be more or less ideal. This is the case for instance when merging requirements expressed by several agents about an artifact to be built or a software to be designed. In that case, the aim of the merging process is to find a consensus between the agents in order to define a description of that ideal world on which the agents agree. But the merging process also depends on the meta-information about the sources. For instance, in the case of merging beliefs provided by several sources, if the respective reliability of the sources is known (in a quantitative setting or in a qualitative setting), it obviously must be used in the merging process: the more reliable a source is, the more we trust it. In the case of requirement merging, if the respective importance of the agents that provide the requirements is known, it also must be used in the merging process: the more important an agent is, the more the result of the merging must agree it. But, if this meta-information is not known, some other types of merging processes must be defined.

Konieczny and Pino-Pérez's [KPP98], [KPP99] address this last case since they do not assume a priority order between the sources to be merged. They define two kinds of merging operators respectively called majority merging operators and arbitration merging operators. The first ones aim at implementing a kind of majority vote between the sources, and the second ones aim at reaching a consensus between the sources by trying to satisfy as much as possible all of them. Konieczny and Pino-Pérez define these two families of merging operators from a semantical point of view. Our work takes as a starting point one of the majority merging operator that have been semantically characterized by the previous work. Our first aim is to define a logic (language, model theory and proof theory) that allows us to reason with data provided by several sources according to that majority operator. Our second aim is to apply these results in the database context and to specify a query evaluator which answers queries addressed to several databases, according to a majority approach. This present paper presents these results, which have been partially presented in [CG01]and [CG02]. It is organized as follows.

In section 2, we focus on the case when the information sources do not contain disjunctions and we present a propositional modal logic called MF

(Majority Fusion). Its language allows us to speak about the content of the information sources (we will say what the information sources believe). Its semantics is a Kripke-type semantics.

In section 3, we prove that this logic effectively axiomatizes a majority merging operator.

Section 4 presents a prover, defined as a meta-program of a PROLOG-type interpreter, which allows one to automatically deduce the data contained in the merged information sources.

Section 5 presents the application of this prover in the context of first-order databases. We specify a query evaluator which answers queries addressed to several databases. Databases we consider are made of an extensional part (i.e. a set of positive or negative facts) and an intensional part (i.e. a set of function-free clauses). A restriction is imposed to these databases in order to avoid disjunctive data.

Extensions to this work are discussed in section 6.

2 The propositional logic MF

2.1 Preliminaries

The semantics of MF logic is based on multi-sets of worlds. So, we recall here some definitions about multi-sets.

Definition 1. A multi-set is a set where redundant occurrences are accepted. Let $MS_1 = [S_1, ..., S_n]$ and $MS_2 = [S_{n+1}, ..., S_m]$ be two multi-sets. The union of two multi-sets is defined by: $MS_1 \bigsqcup MS_2 = [S_1, ..., S_m]$. The membership relation is defined by: $S \in^i MS$ iff there are exactly i occurrences of S in the multi-set MS. Notice that, in the limit case, $S \in^0 MS$ iff there is no occurrence of S in MS, i.e. $S \notin MS$.

Let us then introduce a notation that will be used in the rest of the paper.

Notations. If db and db' denote two information sources, then $db * db'$ will denote the information source obtained by merging db and db'[1]. By information source we mean any information source to be merged (in that case, we call it primitive) and also any information source obtained after merging some information sources.

Example. For instance, if we face three (primitive) information sources db_1, db_2 and db_3 then $db_1 * db_2$, and $(db_1 * db_2) * db_3$ are information sources but they are not primitive. The first one denotes the one obtained by merging db_1 and db_2. The second one denotes the one obtained by merging $db_1 * db_2$ and db_3.

2.2 MF language

Let us call L the propositional language used to describe the contents of the information sources to be merged. The language L' of logic MF is obtained

[1] At this step, this binary operation is not formally defined but it will be in the following, in the model theory and the proof theory as well.

from L by adding several modal operators of the following form: B^i_{db} and B_{db}, where i is an integer and db denotes an information source (primitive or not).

We expect that the formula $B^i_{db}l$ means that there are exactly i occurrences of the literal l in db. And we expect that the formula $B_{db}F$ means that the information source db believes F.

Informally speaking, we introduce the modalities B^i_{db} for being able to count the occurrences of a literal in an information source. The idea is that, when merging two information sources, the number of occurrences of a literal is the sum of the numbers of its occurrences in the two information sources respectively. Then we want that a literal is believed by an information source if the number of occurrences of that literal is strictly greater than the number of occurrences of its negation.

The formal definition of L' is the following:

Definition 2. If F is a formula of L and if B^i_{db} and B_{db} are modal operators, then $B^i_{db}F$ and $B_{db}F$ are formulas of L'. If F_1 and F_2 are formulas of L' then, $\neg F_1$, $F_1 \wedge F_2$ are formulas of L'. $F_1 \vee F_2$ and $F_1 \rightarrow F_2$ are defined from the previous ones as usually.

One can notice that modal operators only govern formulas without modal operators.

Example. For instance, assume that db_1 and db_2 are the two information sources to be merged, then the modal operators we need are: $B^i_{db_1}$, B_{db_1}, $B^i_{db_2}$, B_{db_2}, $B^i_{db_1*db_2}$, $B_{db_1*db_2}$, $B^i_{db_2*db_1}$ and $B_{db_2*db_1}$.

We expect that, for instance: $B^1_{db_1}a$ means that db_1 contains one occurrence of a. $B^0_{db_2}a$ means that db_2 contains no occurrence of a. $B^1_{db_1*db_2}a$ means that the information source obtained by merging db_1 and db_2 contains one occurrence of a. Finally, $B_{db_1*db_2}a$ means that the information source obtained by merging db_1 and db_2 believes a.

2.3 Semantics

The semantics of MF is a Kripke-type one [Che80]. Models are defined by:

Definition 3. Models of MF. A model of MF is a tuple $< W, val, R, B >$ such that:

- W is a set of worlds.
- val is a valuation function[2] which associates any proposition of L with a set of worlds of W.
- R is a set of functions denoted f_{db}, where db is an information source (primitive or not). Each function f_{db} associates any world of W with a multi-set of sets of worlds of W.

[2]It satisfies: $val(P) \neq \emptyset$ iff P is a satisfiable propositional formula, $val(\neg P) = W \setminus val(P)$, $val(P \wedge Q) = val(P) \cap val(Q)$.

- B is a set of functions denoted g_{db}, where db is an information source (primitive or not). Each function g_{db} associates any world of W with a set of worlds of W.

This tuple is constrained by two constraints given below, but before, we need to give the following definition:

Definition 4. Let w and w' be two W worlds. The distance $d(w, w')$ between w and w' is defined by the number of propositional letters p such that $w \in val(p)$ and $w' \notin val(p)$ (that distance is usually called Hamming distance). Let $MS = [S_1 ... S_n]$ be a multi-set of sets of worlds. Then the distance $dsum(w, MS)$ between a world w and MS is defined by : $dsum(w, MS) = \sum_{i=1}^{n} \min_{w' \in S_i} d(w, w')$. Finally, any multi-set of sets of worlds MS is associated with a pre-order $\leq_{MS}$ or W defined by: $w \leq_{MS} w'$ iff $dsum(w, MS) \leq dsum(w', MS)$.

Definition 3 (continued). Models of MF.
The previous tuple $< W, val, R, B >$ is constrained by the two following constraints:

(C1) If db and db' denote two information sources, then:
$\forall w \in W \quad f_{db*db'}(w) = f_{db}(w) \bigsqcup f_{db'}(w)$

(C2) If db is an information source, then $\forall w \in W \quad g_{db}(w) = \min_{\leq_{f_{db}(w)}} W$

The constraint **(C1)** reflects the fact that the occurrences of a literal in the merged information source $db * db'$ are the union of its occurrences in db and of its occurrences in db'. So, it will be the case that the number of occurrences of a literal in $db * db'$ is the sum of the number of its occurrences in db and the number of its occurrences in db',

The constraint **(C2)** expresses that the models of the information source which is obtained by this majority merging operator are the minimal W worlds according to the pre-order $\leq_{f_{db}(w)}$. It will be proved in section 3 that this corresponds to one majority merging operator defined in [KPP98].

Definition 5. Satisfaction of formulas.
Let $M =< W, val, R, B >$ be a model of MF and let $w \in W$. Let p be a propositional letter of L. Let F, F_1 and F_2 be formulas of L' and let db be any information source (primitive or not).

$$
\begin{array}{lll}
M, w \models_{MF} p & \text{iff} & w \in val(p) \\
M, w \models_{MF} \neg F_1 & \text{iff} & M, w \not\models_{MF} F_1 \\
M, w \models_{MF} F_1 \wedge F_2 & \text{iff} & M, w \models_{MF} F_1 \quad \text{and } M, w \models_{MF} F_2 \\
M, w \models_{MF} B^i_{db} F & \text{iff} & val(F) \in^i f_{db}(w) \\
M, w \models_{MF} B_{db} F & \text{iff} & g_{db}(w) \subseteq val(F)
\end{array}
$$

Definition 6. Valid formulas in MF.
Let F be a formula of L'. F is a valid formula in MF iff $\forall M$ model of MF, $\forall w \in W$, $M, w \models_{MF} F$. We note $\models_{MF} F$.

2.4 Proof theory

In the following, db and db' denote information sources (primitive or not), F and G denote formulas of L, l, l_1,..., l_n denote literals of L and i, j, k denote integers.

The axiom schemata of MF are:

($\mathbf{A_0}$) Axiom schemata of propositional logic

($\mathbf{A_1}$) $B_{db}\neg F \rightarrow \neg B_{db}F$

($\mathbf{A_2}$) $B_{db}F \wedge B_{db}(F \rightarrow G) \rightarrow B_{db}G$

($\mathbf{A_3}$) $B^i_{db}l \rightarrow \neg B^j_{db}l$ if $i \neq j$

($\mathbf{A_4}$) $B^i_{db}l \wedge B^j_{db'}l \rightarrow B^k_{db*db'}l$ if $k = i + j$

($\mathbf{A_5}$) $B^i_{db}l \wedge B^j_{db}\neg l \rightarrow B_{db}l$ if $i > j$

($\mathbf{A_6}$) $B^i_{db}l \wedge B^i_{db}\neg l \rightarrow \neg B_{db}l$

($\mathbf{A_7}$) $B_{db}(l_1 \vee ... \vee l_n) \rightarrow B_{db}l_1 \vee ... \vee B_{db}l_n$ with $\forall i \in \{1...n\} \forall j \in \{1...n\}\, l_i \neq \neg l_j$

The inference rules are :

(MP) If $\vdash_{MF} F$ and $\vdash_{MF} (F \rightarrow G)$ then $\vdash_{MF} G$

(Nec) $\vdash_{MF} F$ then $\vdash_{MF} B_{db}F$ for any modality B_{db}.

$\vdash_{MF} F$ denotes as usual, theorems of MF, i.e formulas that are instances of axiom schemata or that can be deduced by using axiom schemata and inference rules.

Modalities B_{db} are belief modalities and are governed by KD axioms,($\mathbf{A_0}$), ($\mathbf{A_1}$), ($\mathbf{A_3}$).

($\mathbf{A_3}$) says that the number of occurrences of a literal in an information source is unique.

($\mathbf{A_4}$) expresses the facts that the number of occurrences of a literal in the merged information source $db * db'$ is the sum of the its occurrences in db and the number of its occurrences in db'.

($\mathbf{A_5}$), ($\mathbf{A_6}$) express the majority aspect of the underlying merging operator. First, a literal l is believed by a source db if the number of its occurrences is strictly greater then the number of the occurrences of its negation. If the number of the occurrences of l is equal to the number of occurrences of its negation, then that literal and its negation are not believed by the information source.

($\mathbf{A_7}$) expresses that if an information source (primitive or not) believes a disjunction of literals, which is not a tautology, then it believes at least one of its literals.

Excluding disjunctions which are tautologies is necessary since due to inference rule (Nec), any tautology is believed. So, for instance $a \vee \neg a$ is believed (due to Nec) even though neither a nor $\neg a$ is believed. (A7) prevents the case when a database believes, for instance $a \vee b$ and does not believe a nor b. This comes to restrict the information sources we consider to sets of literals.

2.5 Soundness and completeness for some interesting formulas

Definition 7. Let $db_1 ... db_n$ be n finite sets of literals to be merged, each of them being consistent. We define the formula ψ by:

$$\psi = \bigwedge_{i=1}^{n} (\bigwedge_{l \in db_i} B^1_{db_i} l \wedge \bigwedge_{l \notin db_i} B^0_{db_i} l)$$

ψ lists the information we have about the content of the given sources to be merged. More precisely, it expresses that each literal it contains has one and only one occurrence in it, and that each literal it does not contain has no occurrence in it.

The following result proves that the model theory and the proof theory previously presented are equivalent for formulas of the form $\psi \rightarrow B_{db}F$, where db is any information source.

Proposition 1. Let ψ be the formula defined by definition 7. Let F be a formula of L and db an information source (primitive or not). Then we have:
$\models_{MF} \psi \rightarrow B_{db}F \iff \vdash_{MF} \psi \rightarrow B_{db}F$ and
$\models_{MF} \psi \rightarrow \neg B_{db}F \iff \vdash_{MF} \psi \rightarrow \neg B_{db}F$
The proof is given in the appendix

Proposition 2. Let ψ be the formula defined by definition 7. Let F be a formula of L and db an information source (primitive or not). Then:

$$\not\vdash_{MF} \psi \rightarrow B_{db}F \iff \vdash_{MF} \psi \rightarrow \neg B_{db}F$$

The proof is given in the appendix

2.6 Example

Here, we give some examples of proofs in MF logic.

We consider three information sources: $db_1 = \{a, b\}$, $db_2 = \{a, \neg c\}$, $db_3 = \{\neg a, c\}$. By definition 7, ψ is: $B^1_{db_1} a \wedge B^1_{db_1} b \wedge B^0_{db_1} c \wedge B^0_{db_1} \neg c \wedge B^0_{db_1} \neg a \wedge B^0_{db_1} \neg b \wedge B^1_{db_2} a \wedge B^1_{db_2} \neg c \wedge B^0_{db_2} b \wedge B^0_{db_2} \neg b \wedge B^0_{db_2} \neg a \wedge B^0_{db_2} c \wedge B^1_{db_3} \neg a \wedge B^1_{db_3} c \wedge B^0_{db_3} b \wedge B^0_{db_3} \neg b \wedge B^0_{db_3} a \wedge B^0_{db_3} \neg c$

Here are some theorems of MF we can derive:

(α) $\vdash \psi \rightarrow B^2_{db_1 * db_2} a$ (by (A_4))

(β) $\vdash \psi \rightarrow B^2_{(db_1 * db_2) * db_3} a$ (by (α) and (A_4))

(γ) $\vdash \psi \rightarrow B^0_{db_1 * db_2} \neg a$ (by (A_4))

(δ) $\vdash \psi \rightarrow B^1_{(db_1 * db_2) * db_3} \neg a$ (by (γ) and (A_4))

Thus, finally, from (β), (δ) and (A_5), we can prove:

(ζ) $\vdash \psi \rightarrow B_{(db_1 * db_2) * db_3} a$

This theorem means that the information source obtained by merging db_1, db_2 and db_3 believes a. Notice that this illustrates a majority attitude, since two primitive information sources believe a while only one believes $\neg a$.

In the same way, we prove:

(η) $\quad \vdash \psi \rightarrow B_{(db_1 * db_2) * db_3} b$

Thus, from (ζ), (η), (A_0) and (A_2) we prove:

$\vdash \psi \rightarrow B_{(db_1 * db_2) * db_3}(a \wedge b)$

This theorem means that the information source obtained by merging db_1, db_2 and db_3 believes $(a \wedge b)$.

Similarly, we have:

$\theta : \vdash_{MF} \psi \rightarrow B^0_{db_1 * db_2} c$ (by (A_4))

$\iota : \vdash_{MF} \psi \rightarrow B^1_{(db_1 * db_2) * db_3} c$ (by (θ) and (A_4))

$\kappa : \vdash_{MF} \psi \rightarrow B^1_{db_1 * db_2} \neg c$ (by (A_4))

$\lambda : \vdash_{MF} \psi \rightarrow B^1_{(db_1 * db_2) * db_3} \neg c$ (by (θ) and (A_4))

$\mu : \vdash_{MF} \psi \rightarrow \neg B_{(db_1 * db_2) * db_3} c$ (by (ι), (λ) and (A_6))

$\nu : \vdash_{MF} \psi \rightarrow \neg B_{(db_1 * db_2) * db_3} \neg c$ (by (ι), (λ) and (A_6))

Thus, finally, by (μ), (ν) and (A_0) we can prove:

$\vdash_{MF} \psi \rightarrow \neg B_{(db_1 * db_2) * db_3} c \wedge \neg B_{(db_1 * db_2) * db_3} \neg c$

This theorem means that the information source obtained by merging db_1, db_2 and db_3 does not believe c nor $\neg c$.

3 Relation with Konieczny and Pino-Pérez's work

In this section, we formally prove that MF logic allows one to reason with merged data obtained by a majority operator. More specifically, we focus on one majority merging operator defined by Konieczny and Pino-Pérez. And we establish a relation between some theorems of MF and the information source obtained by that operator.

First, let us recall the definition introduced by Konieczny and Pino-Pérez[3]

Let $db_1 ... db_n$ be n information sources to be merged.

Konieczny and Pino-Pérez define a majority merging operator, denoted Δ_Σ, such that the models of the information source which is obtained from merging db_1 ... db_n with this operator, is semantically characterized by:

$$Mod(\Delta_\Sigma([db_1, ..., db_n])) = \min_{\leq^\Sigma_{[db_1 ... db_n]}} \quad \text{(W)}$$

where W denotes the set of all the interpretations of the language L (the propositional language used to describe the contents of the informations sources). $\leq^\Sigma_{[db_1 ... db_n]}$ is a total pre-order on W defined by:

$$w \leq^\Sigma_{[db_1 ... db_n]} w' \text{ iff } d_\Sigma(w, [db_1 ... db_n]) \leq d_\Sigma(w', [db_1 ... db_n])$$

with

$$d_\Sigma(w, [db_1 ... db_n]) = \sum_{i=1}^{n} \min_{w' \in Mod(db_i)} d(w, w')$$

[3]One will notice that we slightly change the presentations of these definitions to remain coherent with what has already been presented.

where $Mod(db_i)$ is the set models of db_i and $d(w, w')$ is the Hamming distance.

In other words, when merging $db_1 ... db_n$ with the operator Δ_Σ, the result is semantically characterized by the interpretations which are minimal according to the pre-order $\leq^{\Sigma}_{[db_1,...,db_n]}$.

The following proposition establishes the relation between some theorems of logic MF and the result of this majority merging operator.

Proposition 3. Let $db_1 ... db_n$ be n finite and consistent sets of literals to be merged and F be a formula of L. With the notations previously introduced, we have:

$$\vdash \psi \rightarrow B_{(...(db_1 * db_2) * ... db_n)} F \Longleftrightarrow \Delta_\Sigma([db_1 ... db_n]) \models F$$

The proof is given in the appendix

In other words, the information source whose beliefs are characterized by theorems $\vdash \psi \rightarrow B_{(...(db_1 * db_2) * ... db_n)} F$, is equivalent to $\Delta_\Sigma([db_1 ... db_n])$.

This proves that logic MF is a logic for reasoning with merged data obtained by a majority merging operator, when information sources are sets of literals.

4 Automated deduction in MF

In this section, we deal with implementation aspects. We present a theorem prover for logic MF. It allows one to answer questions of the form: "given the description of the information source contents, is the atomic formula F deducible after merging them?" i.e. it allows one to prove theorems of the form: $\psi \rightarrow B_{db} F$. Extension to non-atomic formulas will be discussed in section 6.

One will notice that in this prover, the formula ψ introduced previously will not be used. Indeed, ψ was introduced for theoretical reasons. Its aim was to describe in extension what is believed and what is not believed in the primitive information sources. But in the prover, we will only need to list the explicit beliefs of the sources. Propositions which are not believed will be derived by negation as failure.

4.1 The meta-language

Let us consider a meta-language ML, based on language L, defined by:

- constants symbols of ML are propositional letters of L, names of informations sources plus a constant symbol denoted *nil* and constants denoting integers: 1, 2, etc.
- a binary function noted $*$. By convention, $(db_{i_1} * ... * db_{i_k})$ represents the term: $db_{i_1} * (db_{i_2} ... * (db_{i_k} * nil)...)$. This function will be used to denote the information sources obtained by merging information sources $db_{i_1} ... db_{i_k}$.

- a binary function denoted + which is the sum of integers.
- a unary function symbol $\neg$. By convention, $\neg l$ represents the term $\neg(l)$. This function will be used to describe the object-level negation.
- the binary meta-predicate symbols are: B_{exp}, B, $=$ and $>$
- A ternary meta-predicate symbol is R.
- A unary meta-predicate symbol is NIL.

The intuitive semantics of the predicates is the following:
- $B_{exp}(db, l)$ is true if literal l is explicitly stored in the primitive information source db.
- $R(db, l, i)$ is true if l appears i times in the information source db.
- $B(db, l)$ is true if the information source db believes l.
- $NIL(db)$ is true if db is *nil*.
- $i = j$ (resp, $(i > j)$) is true if integers i and j are equal (resp, if integer i is strictly greater than integer j). These two predicates will be defined in extension, in the meta-program, by a finite number of facts.

4.2 The meta-program

If there are n information sources to be merged $db_1, \ldots, db_n$, then let META be the following set of the ML formulas:

(1) $B_{exp}(db, l)$ if the literal l belongs to the primitive information source db
(2) $\neg NIL(db_2) \wedge R(db_1, l, i) \wedge R(db_2, l, j) \wedge (k = i + j) \rightarrow R(db_1 * db_2, l, k)$
(3) $NIL(db_2) \wedge B_{exp}(db_1, l) \rightarrow R(db_1 * db_2, l, 1)$[4]
(4) $NIL(db_2) \wedge \neg B_{exp}(db_1, l) \rightarrow R(db_1 * db_2, l, 0)$
(5) $R(db, l, i) \wedge R(db, \neg l, j) \wedge (i > j) \rightarrow B(db, l)$
(6) $NIL(nil)$
(7) $k = (r + l)$ and $(r + l) = k$ for any k in $\{1...n\}$ for any r in $\{1...k\}$ and for any l such that $l = k - r$
(8) $k > r$ for any k in $\{1...n\}$ and for any r in $\{1...k - 1\}$

Notice that there is a finite number of axioms (7) and axioms (8).

The following result ensures the correctness of this meta program.

Proposition 4. Let l be a literal, let db denoting an information source (primitive or not). Then, using negation-as-failure on the meta-program META,
(1) PROLOG succeeds in proving $B(db, l)$ iff $\models_{MF} (\psi \rightarrow B_{db} l)$
(2) PROLOG fails in proving $B(db, l)$ iff $\models_{MF} (\psi \rightarrow \neg B_{db}\ l)$
The proof is given in the appendix

[4] Recall that primitive sources are sets of literals so each literal which belongs to a source has exactly one occurrence in it.

5 Application to multi-databases: specification of a query evaluator

In this section we apply the previous results for specifying a query evaluator which answers queries addressed to several databases. But in order to export these results to first order databases, we consider only databases which are "equivalent to sets of ground literals". Such databases are defined below.

5.1 Databases equivalent to sets of ground literals

Let LO be a first order language.

Definition 8 . A database is a pair $DB =< EDB, IDB >$[5] such that EDB is a non empty and finite set of positive or negative ground literals of LO and IDB is a finite and consistent set of clauses of LO written without function symbol.

Notice that literals in EDB can be positive or negative.

Definition 9. Let $DB =< EDB, IDB >$ a database. Let $a_1...a_n$ (resp. $P_1...P_k$) be the constant (resp. predicate) symbols which appear in the formulas of $EDB \cup IDB$. The Herbrand base is the set of positive literals written with the P_i and the a_j. A Herbrand interpretation of DB is an interpretation whose domain is $\{a_1, ..., a_n\}$. A Herbrand model of DB is a Herbrand interpretation which satisfies $EDB \cup IDB$.

Definition 10. Let $HM_1...HM_n$ be the Herbrand models of $EDB \cup IDB$.

Let $L = \{l : l$ is a literal of the Herbrand base such that $\exists\ HM_i \exists\ HM_j$ $HM_i \models l$ and $HM_j \models \neg l\}$

The database $DB =< EDB, IDB >$ is equivalent to a set of ground literals iff for any satisfiable conjunction $l_1 \wedge ... \wedge l_m$ where $\forall i \in \{1...m\}$ such that $l_i \in L$ or $\neg l_i \in L$, there exists HM_{i_0} such that $HM_{i_0} \models l_1 \wedge ... \wedge l_m$.

Example. Consider $DB_1 =< EDB_1, IDB_1 >$ with $EDB_1 = \{p(a)\}$ and $IDB_1 = \{\neg p(x) \vee \neg q(x), p(x) \vee r(x)\}$. The Herbrand models of DB_1 are[6]: $\{p(a)\}$ and $\{p(a), r(a)\}$. We have: $L = \{r(a)\}$ We can check that $\neg r(a)$ is satisfied in the first Herbrand model and that $r(a)$ is satisfied in the second. So DB_1 is equivalent to a set of ground literals.

Consider now $DB_2 =< EDB_2, IDB_2 >$ with $EDB_2 = \{p(a)\}$ and $IDB_2 = \{\neg p(x) \vee q(x) \vee r(x)\}$. The Herbrand models of DB_2 are $\{p(a), q(a)\}$, $\{p(a), r(a)\}$ and $\{p(a), q(a), r(a)\}$. We have $L = \{r(a), q(a)\}$. We can check that none of the Herbrand models satisfy $\neg q(a) \wedge \neg r(a)$. Thus DB_2 is not equivalent to a set of ground literals.

Proposition 5. Let $DB =< EDB, IDB >$ a database which is equivalent to a set of ground literals. Let $l_1...l_n$ be some ground literals of LO such that $l_1 \vee ... \vee l_n$ is not a tautology. Then,

[5] "EDB" stands for "extensional database" and "IDB" stands for "intensional database".

[6] A model is denoted here by the set of its positive facts.

$EDB \cup IDB \models l_1 \vee ... \vee l_n$ iff $\exists i_0 \in \{1...n\}$ $EDB \cup IDB \models l_{i_0}$
The proof is given in the appendix

This result ensures that, in a database equivalent to a set of ground literals, a disjunction of ground literals which is not a tautology is deducible from the database iff one of these literals is deducible from the database. This implies that there is no real disjunctive data deducible from these databases.

5.2 Specification of a query evaluator

In this section, we use the meta-program defined in section 4 in order to specify a query evaluator which answers queries addressed to several databases. The answers computed by the evaluator are the same that could be computed by a classical evaluator when the query is addressed to the database obtained by merging several databases according to a majority attitude. However, it must be noticed that the databases merging is never computed.

The meta-program defined in section 4 assumes that the information sources are sets of positive or negative propositional literals. Considering only databases which are equivalent to sets of ground literals will allow us to re-use that meta-program: each ground literal will be consider as a propositional one. However, we must extend the meta-program in order to take the clauses of IDB into account.

Extension of the meta-program to take IDB into account:

Let us denote by h the function which associates any clause of IDB with a set of formulas in the following way:

$h(l_1 \vee ... \vee l_n) = \{(\neg l_1 \wedge ... \neg l_{i-1} \wedge \neg l_{i+1} ... \wedge \neg l_n) \rightarrow l_i, \quad i \in \{1, ...n\}\}$

Then, the axiom (1) of the meta-program is replaced by the following ones:

(1.1) $EDB(db, l)$ if the ground literal l is in the EDB part of the database db.
(1.2) $IDB(db, f)$ if the formula f is in h(c), where c is a clause in the IDB part of db.
(1.3) $EDB(db, l) \rightarrow B_{exp}(db, l)$
(1.4) $IDB(db, (r \rightarrow l)) \wedge B_{conj}(db, r) \rightarrow B_{exp}(db, l)$
(1.5) $B_{conj}(db, nil)$
(1.6) $B_{exp}(db, l_1) \wedge B_{conj}(db, r_1) \rightarrow B_{conj}(db, l_1 \wedge r_1)$

Proposition 6. Let $db =< EDB, IDB >$ be such that IDB is not recursive. Let l be a ground literal. Then PROLOG proves $B_{exp}(db, l)$ iff $EDB \cup IDB \models l$.
The proof is given in the appendix

This result ensures that, if IDB is not recursive, axiom (1) can be replaced by axioms (1.1)...(1.6). Thus, using proposition 2, if IDB is not recursive, the meta-program defined for informations sources which are sets of propositional

literals can be used in the case of first order databases which are equivalent to sets of ground literals.

5.2.1 Definition of answers

Let $db_1 ... db_n$ be n databases, each of them being equivalent to a set of literals.

Closed queries Let F be a ground literal. The answer to the query "Is F true in the database obtained by merging $db_1 ... db_n$?" is defined by:
$answer((db_1 * \ldots * db_n), F) = YES$ iff PROLOG proves $B((db_1 * \ldots * db_n), F)$
$answer((db_1 * \ldots * db_n), F) = NO$ iff PROLOG proves $B((db_1 * \ldots * db_n, \neg F)$
$answer((db_1 * \ldots * db_n), F) =?$ else

Notice that we are in an open world approach. That is, in any database (primitive or obtained by merging some databases) it may exist an atomic formula F such that neither F nor $\neg F$ are deducible. This explains why the answer ? is sometimes provided. From an operational point of view, this corresponds to the case when the two goals $B((db_1 * \ldots * db_n), F)$ and $B((db_1 * \ldots * db_n), \neg F)$ finitely fail in PROLOG.

Open queries. Let $db_1 ... db_n$ be n databases and $F(X)$ be an open literal. The answer to the query "What are the X which satisfy F in the database obtained by merging $db_1 ... db_n$", is defined by:
$answer((db_1 * ... * db_n), F(X)) = \{A$: tuple of constant symbols such that PROLOG proves $B((db_1 * ... * db_n), F(A))\}$

5.3 Example

Let us consider the three following databases:

$$db_1 =< EDB_1, IDB >, db_2 =< EDB_2, IDB >, db_3 =< EDB_3, IDB >$$

with:

$EDB_1 = \{student(John), employee(Louis), self(Philip), self(Donald), disabled(Louis)\}$
$EDB_2 = \{employee(Philip), student(Louis), restaurant(John), restaurant(Henry)\}$
$EDB_3 = \{student(John), employee(Philip)\}$
$IDB = \{\forall x \ \ student(x) \rightarrow self(x), \forall x \ \ employee(x) \rightarrow restaurant(x), \forall x \ \ \neg self(x) \vee \neg restaurant(x), \forall x \ \ disabled(x) \rightarrow parking(x), \forall x \ \ employee(x) \rightarrow parking(x), \forall x \ \ student(x) \rightarrow \neg parking(x)\}$

Formulas of IDB express that students eat in the self-service ; employees eat in the restaurant ; nobody can eat in both ; disabled persons and employees can park their car in the parking ; students cannot.

One can notice that each database is equivalent to a set of ground literals (here all the literals are positive) and that IDB is not recursive. Here are some queries and the answers generated by the query evaluator:

Is John a student in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), student(John)) = YES$
This means that, when merging the three databases, we can conclude that $John$ is a student.

Is John not an employee in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), \neg employee(John)) = YES$
This means that, when merging the three databases, we can conclude that $John$ is not an employee.

Is Donald a student in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), student(Donald)) =?$
This means that, when merging the three databases, we cannot prove that $Donald$ is a student, nor that he is not a student.

Who is student in the database obtained by merging db_1 and db_2 ?
$answer((db_1 * db_2), student(x)) = \emptyset$
When considering the first two databases, there is no student.

Who is student in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), student(x)) = \{John\}$
When considering the three databases, $John$ is the only student.

Who is employee in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), employee(x)) = \{Philip\}$
In the database obtained by merging db_1, db_2 and db_3, $Philip$ is the only employee.

Who can park his/her car in the parking in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), parking(x)) = \{Philip\}$
In the database obtained by merging db_1, db_2 and db_3, $Philip$ is the only one who can park his car in the parking.

One can notice that, when merging the three databases, we cannot prove that $Louis$ is an employee nor that he can park in the parking. This illustrates the fact that the majority merging does not take into account the number of proofs of a literal in a given database but it takes into account the number of sources that support it, whatever the "strentgth" with which it supports it. Here, one database supports that $Louis$ can park in the parking (even if it supports twice) and one database supports that $Louis$ cannot. Majority merging cannot decide.

Who goes to the self service in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), self(x)) = \{John, Donald\}$
In the database obtained by merging db_1, db_2 and db_3, $John$ and $Donald$ are the only persons who go to the self service.

Who goes to the restaurant in the database obtained by merging db_1, db_2 and db_3 ?
$answer((db_1 * db_2 * db_3), restaurant(x)) = \{Philip, Henry\}$
In the database obtained by merging db_1, db_2 and db_3, $Philip$ and $Henry$ are the only persons who go to the restaurant.

Is John a student in the database obtained by merging db_1 and db_2 ?
$answer((db_1 * db_2), student(John)) =?$
When considering the first two databases, we cannot prove that $John$ is a student nor that he is not a student.

6 Concluding remarks

Several remarks can be done concerning the previous query evaluator.

First of all, let us say that this query evaluator has been implemented in a PROLOG interpretor written in LISP.

Secondly, we insist on the fact that the merging of the databases is only virtual i.e. the merging of the databases is never computed for answering questions. This implies, for instance, that the user may address a query to db_1, db_2 and db_3 and latter on address a query to db_2 and db_3.

Thirdly, even if we have restricted our presentation to atomic queries, it must be noticed that this query evaluator can easily be extended for answering queries which are not atomic. The solution for extending the evaluator to non-atomic queries is the same that has been described in [Cho98b]. This extension allows one to ask queries which are conjunctions of disjunctions (where any disjunction which is a tautology is removed). The meta-language is extended by two meta-functions $\wedge$ and $\vee$ and the evaluator is extended by adding the following meta-axioms:

$C(db, nil)$
$D(db, s) \wedge C(db, c) \rightarrow C(db, d \wedge c)$
$B(db, l) \rightarrow D(db, l \vee d)$
$B(db, d) \rightarrow D(db, l \vee d)$

Thus, if F and $\neg F$ are two closed formulas written as conjunction of disjunctions of literals (where any disjunction which is a tautology is removed) then the question "if F true in the database obtained by merging $db_1 ... db_n$?" is YES, NO or ? depending on the fact that PROLOG proves $C((db_1 * ... * db_n, F)$, $C((db_1 * ... * db_n, \neg F)$, or none of them.

If $F(X)$ is an open formula written as a conjunction of disjunctions of literals (where any disjunction which is a tautology is removed) then the answer to

the question "What are the X which satisfy F, in the database obtained by merging $db_1 ... db_n$" is the set of tuples of constants A such that PROLOG proves $C(db_1 * ... * db_n, F(A))$.

Furthermore, let us say that this work can be easily extended by distinguishing the different cases when something is believed in a merged information source. Indeed, in the previous work, the answer "YES" is provided to a query "is Q true in the merged database (db1 *...*dbn) ?" when the number of databases which believe Q is strictly greater than the number of databases which believe $\neg Q$. The case when some databases believe Q and no database believes $\neg Q$ is not distinguished. In the same way, the answer "?" is provided to the query "is Q true in the merged database (db1 *...*dbn) ?" when there is a no database which believes Q and also when there are as many databases which believe Q and databases which believe $\neg l$.

If one wants to distinguish these different kinds of situations, we suggest to replace the modalities B_{db} by four new modalities: $Unchallanged_{db}$,$Majority_{db}$ $CompleteLack_{db}$ and BI_{db}. $Unchallanged_{db}l$ intends to mean that in db, there are proofs for l but no proof against l ; $Majority_{db}l$ intends to mean that db believes l by majority ; $CompleteLack_{db}l$ intends to mean that in db, there is a complete lack of information about l and $BI_{db}l$ intends to mean that there is a balanced inconsistency concerning l in db. Furthermore, we suggest to split axioms (A5) and (A6) in two as follows:

(A5.1) $B^i_{db}l \wedge B^0_{db}\neg l \rightarrow Unchallanged_{db}l$ for $i > 0$

(A5.2) $B^i_{db}l \wedge B^j_{db}\neg l \rightarrow Majority_{db}l$ for $i > j > 0$

$(A6.1) B^0_{db}l \wedge B^0_{db}\neg l \rightarrow CompleteLack_{db}l$

$(A6.2) B^i_{db}l \wedge B^i_{db}\neg l \rightarrow BI_{db}l$ for any $i > 0$

As for the query evaluator, the binary meta-predicate B must be replaced by a ternary one and meta-axiom (5) must be replaced by:

$(5.1) R(db, l, 0) \wedge R(db, \neg l, 0) \rightarrow B(CompleteLack, db, l)$

$(5.2) R(db, l, i) \wedge R(db, l, 0) \wedge (i > 0) \rightarrow B(Unchallanged, db, l)$

$(5.3) R(db, l, i) \wedge R(db, \neg l, i) \wedge (i > 0) \rightarrow B(BalancedInconsistency, db, l)$

$(5.4) R(db, l, i) \wedge R(db, \neg l, j) \wedge (i > j) \wedge (j > 0) \rightarrow B(Majority, db, l)$

The answer to a closed query can now be : "Don't know (complete lack of information)", "Don't know (balanced inconsistency)", "YES (there is no proof against it)", "YES (by majority)"or "NO (there is no proof against it)", "NO (by majority)". The answer of an open query can be a list a typed tuples of constants. Running this on the example 5.3 now leads to :

$answer((db_1 * db_2 * db_3), student(John)) = YES$ $(by\ majority)$

$answer((db_1 * db_2 * db_3), \neg employee(John)) = YES$ $(\ there\ is\ no\ proof\ against\ it)$

$answer((db_1 * db_2 * db_3), student(Donald)) = Don'tknow$ $(complete\ lack\ of\ information)$

$answer((db_1 * db_2), student(x)) = \emptyset$

$answer((db_1 * db_2 * db_3), student(x)) = \{John\ (by\ majority)\}$

$answer((db_1 * db_2 * db_3), employee(x)) = \{Philip\ (by\ majority)\}$

$answer((db_1 * db_2 * db_3), parking(x)) = \{Philip\ (there\ is\ no\ proof\ against\ it)\}$

$answer((db_1 * db_2 * db_3), self(x)) = \{John\ (by\ majority), Donald\ (there\ is\ no\ proof\ against\ it)\}$

$answer((db_1 * db_2 * db_3), restaurant(x)) = \{Philip\ (by\ majority), Henry\ (by\ majority)\}$

$answer((db_1 * db_2), student(John)) = Don't\ know\ (balanced\ inconsistency)$

That query evaluator has also been implemented.

Finally, let us say that extending this evaluator in the case when databases contain disjunctive data is still an open question. The model theory of the logic MF can easily be extended when information sources contain disjunctions. However, we must admit that we have not yet found a complete proof-theory corresponding to this model theory nor a correct prover.

Acknowledgements. The authors would like to thank the anonymous referees for contributing helpful comments and suggestions.

References

[BKMS91] C. Baral, S. Kraus, J. Minker, and V.S. Subrahmanian. Combining multiple knowledge bases. *IEEE Trans. on Knowledge and Data Engineering*, 3(2), 1991.

[BKMS92] C. Baral, S. Kraus, J. Minker, and V.S. Subrahmanian. Combining knowledge bases consisting of first order theories. *Computational Intelligence*, 8(1), 1992.

[CG01] L. Cholvy and Ch. Garion. A logic to reason an contradictory beliefs with a majority approach. In *Proceedings of the IJCAI'01 Workshop: Inconsistencies in Data and Knowledge*, Seattle, August 2001.

[CG02] L. Cholvy and Ch. Garion. Anwering queries addressed to merged databases: a query evaluator which implements a majority approach. In *Proceedings of the* 13^{th} *International Symposium on Methodologies for Intelligent Systems*, Lyon, France, June 2002.

[Che80] B. F. Chellas. *Modal logic, an introduction.* Cambridge University Press, 1980.

[Cho93] L. Cholvy. Proving theorems in a multi-sources environment. In *Proceedings of IJCAI'93*, pages 66–71, Chambéry, France, 1993.

[Cho98a] L. Cholvy. Reasoning about merged information. In *Handbook of defeasible reasoning and uncertainty management*, volume 1. Kluwer Academic Publishers, 1998.

[Cho98b] L. Cholvy. Reasoning with data provided by federated databases. *Journal of Intelligent Information Systems*, 10(1), 1998.

[KPP98] S. Konieczny and R. Pino-Pérez. On the logic of merging. In *Proceedings of KR'98*, pages 488-498, Trento, 1998.

[KPP99] S. Konieczny and R. Pino-Pérez. Merging with integrity constraints. In *Proceedings of the Fifth European Conference on Symbolic and SQuantitative Appoaches to Reasoning with Uncertainty (ESCQARU'99)*, volume 1638 of *Lecture Notes in Artificial Intelligence*, pages 233-244. Springer-Verlag, 1999.

[Lia00] C. Liau. A conservative approach to distributed belief fusion. In *Proceedings of* 3^{rd} *International Conference on Information Fusion (FUSION)*, 2000.

[Lin96] J.. Lin. Integration of weighted knowldege bases. *Artificial Intelligence*, 83:363–378, 1996.

[LM98] J. Lin and A.O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information Systems*, 7(1), 1998.

[SDL+98] S.Benferhat, D. Dubois, J. Lang, H. Prade, A. Saffiotti, and P. Smets. A general approach for inconsistency handling and merging information in prioritized knowledge bases. In *Proc. of KR'98*, Trento, 1998.

[Sub94] V.S. Subrahmanian. Amalgamating knowledge bases. *ACM Transactions on Database Systems*, 19(2):291–331, 1994.

Appendix: proofs

Proposition 1. Let ψ be the formula defined by definition 7. Let F be a formula of L and db an information source. Then we have:
$\models_{MF} \psi \rightarrow B_{db}F \iff \vdash_{MF} \psi \rightarrow B_{db}F$ and
$\models_{MF} \psi \rightarrow \neg B_{db}F \iff \vdash_{MF} \psi \rightarrow \neg B_{db}F$

Proof.
$\Longleftarrow$). We prove the soundness of the proof theory for any formula of L', by proving, as usual, that instances of the axiom schemas are valid formulas and that inference rules preserve the validity.

$\Longrightarrow$). We first prove the four following lemmas :
Lemma 1.1. Let $M =< W, val, R, B >$ be a model in which ψ is satisfied. Then, for any primitive information source db_i we have : $f_{db_i} = \{val(l) : l \in db_i\}$

Lemma 1.2. Let $M =< W, val, R, B >$ be a model in which ψ is satisfied. Then, for any primitive information source db_i, for any literals l and for any $w \in W$ we have : $val(l) \in f_{db_i}(w) \Longrightarrow val(\neg l) \notin f_{db_i}(w)$.

Lemma 1.3 Let MS be a multi-set of valuations of literals such that for any literal l, $val(l) \in MS \Longrightarrow val(\neg l) \notin MS$. Then $\min_{\leq_{MS}}(W) = val(\bigwedge_{val(l) \in MS} l)$.

Lemma 1.4 Let l be a literal,i an integer and db an information source, primitive or not. We have: $\models_{MF} \psi \rightarrow B^i_{db}l \Longrightarrow \vdash_{MF} \psi \rightarrow B^i_{db}l$ and $\models_{MF} \psi \rightarrow \neg B^i_{db}l \Longrightarrow \vdash_{MF} \psi \rightarrow \neg B^i_{db}l$.

With these lemmas, we can now prove the ($\Longrightarrow$) part. For doing so, we define for any $n \geq 1$:
Definition $\mathcal{H}(n)$: for any formula F

$$\models_{MF} \psi \rightarrow B_{db_1 * \ldots * db_n}F \Longrightarrow \vdash_{MF} \psi \rightarrow B_{db_1 * \ldots * db_n}$$

$$\models_{MF} \psi \rightarrow \neg B_{db_1 * \ldots * db_n}F \Longrightarrow \vdash_{MF} \psi \rightarrow \neg B_{db_1 * \ldots * db_n}F$$

where $\forall i \in \{1 \ldots n\}$ db_i is a primitive information source.
The proof by induction of $\mathcal{H}(n)$ for all $n \geq 1$ proves the completeness property.

Proof of $\mathcal{H}(1)$
Let db be a consistent primitive information source.

1. if $\models_{MF} \psi \rightarrow B_{db}F$ then $\models_{MF} \psi \rightarrow B_{db}F \Rightarrow \forall M \forall w \;\; M, w \models_{MF} \psi \rightarrow B_{db}F$, so $\forall M \forall w \;\; M, w \models_{MF} \psi \;\Rightarrow\; M, w \models_{MF} B_{db}F$.

   Let M be a MF-model which satisfies ψ. By lemma 1.1, $\forall w \in W \; f_{db}(w) = \{val(l) \;\; : \;\; l \in db\}$. Let w be a world of W. db is consistent, so

by lemma 1.2, $val(l) \in f_{db}(w) \Rightarrow val(\neg l) \notin f_{db}(w)$. Thus, by lemma 1.3, $\min_{\leq_{f_{db}(w)}} W = val(\bigwedge_{val(l)\in f_{db}(w)} l)$. But $g_{db}(w) = \min_{\leq_{f_{db}(w)}} W$, so $g_{db}(w) = val(\bigwedge_{val(l)\in f_{db}(w)} l)$.

Moreover, $M, w \models_{MF} B_{db}F \Rightarrow g_{db}(w) \subset val(F)$, so $val(\bigwedge_{val(l)\in f_{db}(w)} l) \subset val(F)$ and $\models_{MF} \bigwedge_{l\in db} l \rightarrow F$.

(a) if F is a literal l', then $\models_{MF} \bigwedge_{l\in db} l \rightarrow l'$, so $l' \in db$.

$$\begin{array}{rcl} l' \in db & \Rightarrow & \neg l' \notin db \text{ because } db \text{ is consistent} \\ & \Rightarrow & \vdash_{MF} \psi \rightarrow B^1_{db} l' \text{ and} \\ & & \vdash_{MF} \psi \rightarrow B^0_{db} \neg l' \\ \text{axiom schemata } \mathbf{A4} & \Rightarrow & \psi \rightarrow B_{db} l' \end{array}$$

(b) if $F \equiv l_1 \vee \ldots \vee l_m$ then if $F \equiv \top$, then $\vdash_{MF} F$, so $\vdash_{MF} B_{db}F$, thus $\vdash_{MF} \psi \rightarrow B_{db}F$.
If $F \not\equiv \top$, then $\models_{MF} \bigwedge_{l\in db} l \rightarrow l_1 \vee \ldots \vee l_m$. Therefore, there is $k \in \{1 \ldots m\}$ so that $\models_{MF} \bigwedge_{l\in db} l \rightarrow l_k$. This case is the same as case 1.a.

(c) if F is a conjunction of disjunctions, that is to say $F \equiv \bigwedge_{k\in\{1\ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$, then $\models_{MF} \bigwedge_{l\in db} l \rightarrow \bigwedge_{k\in\{1\ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$ so $\forall k \in \{1 \ldots n\}$ $\models_{MF} \bigwedge_{l\in db} l \rightarrow l_{k_1} \vee \ldots \vee l_{k_m}$. This case is the same as case 1.b, because of the KD property of B_{db}.

2. if $\models_{MF} \psi \rightarrow \neg B_{db}F$, we prove $g_{db}(w) = val(\bigwedge_{l\in db} l)$ for any MF-model M of ψ and any world w of W by the same method.

$$\begin{array}{rcl} M, w \models_{MF} \neg B_{db}F & \models_{MF} \Leftrightarrow & g_{db}(w) \not\subseteq val(F) \\ & \Leftrightarrow & \not\models (\bigwedge_{l\in db} l) \rightarrow F \end{array}$$

(a) if F is a literal l', then $\not\models_{MF} (\bigwedge_{l\in db} l) \rightarrow l'$. In this case, $l' \notin db$ (because db is consistent).

$$l' \notin db \quad \Rightarrow \quad \vdash_{MF} \psi \rightarrow B^0_{db} l'$$

There are two different cases:

- either $\vdash_{MF} \psi \rightarrow B^0_{db}\neg l'$ and then by axiom schemata **A6**, $\vdash_{MF} \psi \rightarrow \neg B_{db} l'$;

- or $\vdash_{MF} \psi \rightarrow B^i_{db}\neg l'$ with $i > 0$ and:

$$\begin{array}{rcl} \text{axiom schemata } \mathbf{A5} & \Rightarrow & \vdash_{MF} \psi \rightarrow B_{db}\neg l' \\ \text{axiom schemata } \mathbf{A1} & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg B_{db} l' \end{array}$$

(b) if $F \equiv l_1 \vee \ldots \vee l_m$ then:

$$\begin{array}{rcl} \not\models (\bigwedge_{l \in db} l) \rightarrow F & \Rightarrow & \forall i \in \{1 \ldots m\}\ l_i \notin db \\ & \Rightarrow & \vdash_{MF} \psi \rightarrow B^0_{db} l_1 \wedge \ldots \wedge B^0_{db} l_m \\ \text{same proof as case 2.a} & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg B_{db} l_1 \wedge \ldots \wedge \neg B_{db} l_m \\ & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg (B_{db} l_1 \vee \ldots \vee B_{db} l_m) \\ \text{axiom schemata } \mathbf{A7} & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg B_{db} (l_1 \vee \ldots \vee l_m) \end{array}$$

(c) if F is a conjunction of clauses, i.e. $F \equiv \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$, then:

$$\begin{array}{rcl} \not\models (\bigwedge_{l \in db} l) \rightarrow F & \Rightarrow & \exists i \in \{1 \ldots n\}\ \not\models_{MF} (\bigwedge_{l \in db} l) \rightarrow (l_{k_1} \vee \ldots \vee l_{k_m}) \\ \text{by the previous proof} & \Rightarrow & \exists i \in \{1 \ldots n\}\ \vdash_{MF} \psi \rightarrow \neg B_{db} (l_{k_1} \vee \ldots \vee l_{k_m}) \\ & \Rightarrow & \vdash_{MF} \psi \rightarrow \bigvee_{k \in \{1 \ldots n\}} \neg B_{db} (l_{k_1} \vee \ldots \vee l_{k_m}) \\ & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg (\bigwedge_{k \in \{1 \ldots n\}} B_{db} (l_{k_1} \vee \ldots \vee l_{k_m})) \\ \text{axiom sch. } \mathbf{A1} \text{ et } \mathbf{A2} & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg B_{db} (\bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})) \end{array}$$

We proved that for all primitive consistent information source db and for all propositional formula F:

$$\begin{array}{rcl} \models_{MF} \psi \rightarrow B_{db} F & \Rightarrow & \vdash_{MF} \psi \rightarrow B_{db} F \\ \models_{MF} \psi \rightarrow \neg B_{db} F & \Rightarrow & \vdash_{MF} \psi \rightarrow \neg B_{db} F \end{array}$$

$\mathcal{H}(1)$ is true.

Proof of $\mathcal{H}(n)$ with $n > 1$

Let db be a information source obtained by merging n primitive information sources ($n \geq 1$) and db_{n+1} a primitive information source.

1. assume that $\models_{MF} \psi \rightarrow B_{db * db_{n+1}} F$.

   (a) if F is a literal l of L then for any MF-model M of ψ and for any world w, $g_{db * db_{n+1}}(w) \subset val(l)$.

Let i and j be the two integers such that $val(l) \in^i f_{db*db_{n+1}}(w)$ and $val(\neg l) \in^j f_{db*db_{n+1}}(w)$. Assume that $i \leq j$. Let w_1 and w_2 be two worlds of W such that $w_1 \models_{MF} \neg l$ and $w_2 \models_{MF} l$.
In this case:

$$f_{db*db_{n+1}}(w) = [\underbrace{val(l)\ldots val(l)}_{i\ times}\ \underbrace{val(\neg l)\ldots val(\neg l)}_{j\ times}\ \underbrace{val(l_1)\ldots val(l_m)}_{\forall k\in\{1\ldots m\}\ l_k\neq l \text{ et } l_k\neq\neg l}\ ]$$

So:

$$\begin{aligned} dsum(w_1, f_{db*db_{n+1}}(w)) &= 0*j + 1*i + \\ &\quad \sum_{k\in\{1\ldots m\}} \min_{w'\in f_{db*db_{n+1}}(w)} d(w_1, w') \\ &= i \\ dsum(w_2, f_{db*db_{n+1}}(w)) &= 1*j + 0*i + \\ &\quad \sum_{k\in\{1\ldots m\}} \min_{w'\in f_{db*db_{n+1}}(w)} d(w_2, w') \\ &= j \end{aligned}$$

Let us suppose that $i \leq j$. There are two different cases:

- $i < j$, then $g_{db*db_{n+1}}(w) \subset val(\neg l)$, so $g_{db*db_{n+1}}(w) \not\subset val(l)$. But $g_{db*db_{n+1}}(w) \subset val(l)$ therefore the hypothesis $i < j$ is false.
- $i = j$, and then $g_{db*db_{n+1}}(w) \not\subset val(l)$ so the hypothesis is false.

So $\exists i\ \exists j\ i > j \geq 0$ such that $val(l) \in^i f_{db*db_{n+1}}(w)$ and $val(\neg l) \in^j f_{db*db_{n+1}}(w)$.
By building of $f_{db*db_{n+1}}(w)$:

- $\exists i_1\ \exists i_2$ such that $i_1 + i_2 = i$, $val(l) \in^{i_1} f_{db}$ and $val(l) \in^{i_2} f_{db_{n+1}}$;
- $\exists j_1\ \exists j_2$ such that $j_1 + j_2 = i$, $val(l) \in^{j_1} f_{db}$ and $val(l) \in^{j_2} f_{db_{n+1}}$;

Therefore:

$$\left\{ \begin{array}{l} \models_{MF} \psi \rightarrow B^{i_1}_{db} l \quad \text{and} \quad \models_{MF} \psi \rightarrow B^{i_2}_{db_{n+1}} l \\ \models_{MF} \psi \rightarrow B^{j_1}_{db} \neg l \quad \text{and} \quad \models_{MF} \psi \rightarrow B^{j_2}_{db_{n+1}} \neg l \end{array} \right.$$

$$\text{lemma 1.4} \quad \Rightarrow \quad \left\{ \begin{array}{l} \vdash_{MF} \psi \rightarrow B^{i_1}_{db} l \quad \text{and} \quad \vdash_{MF} \psi \rightarrow B^{i_2}_{db_{n+1}} l \\ \vdash_{MF} \psi \rightarrow B^{j_1}_{db} \neg l \quad \text{and} \quad \vdash_{MF} \psi \rightarrow B^{j_2}_{db_{n+1}} \neg l \end{array} \right.$$

$$\text{axiom sch. } \mathbf{A4} \quad \Rightarrow \quad \left\{ \begin{array}{l} \vdash_{MF} \psi \rightarrow B^{i}_{db*db_{n+1}} l \\ \vdash_{MF} \psi \rightarrow B^{j}_{db*db_{n+1}} \neg l \end{array} \right.$$

$$\text{axiom sch. } \mathbf{A5} \quad \Rightarrow \quad \{\ \vdash_{MF} \psi \rightarrow B_{db*db_{n+1}} l \text{ (since } i > j)$$

(b) if F is a disjunction of literals, i.e. $F \equiv l_1 \vee \ldots \vee l_m$ then if $F \equiv \top$, $\vdash_{MF} F$, so $\vdash_{MF} B_{db*db_{n+1}} F$ (by (Nec)), and then $\vdash_{MF} \psi \rightarrow B_{db*db_{n+1}} F$.

If $F \not\equiv \top$, then for all $w \in W$ $f_{db*db_{n+1}}(w)$ is such that $val(l) \in f_{db*db_{n+1}}(w) \Longrightarrow val(\neg l) \notin f_{db*db_{n+1}}(w)$. So, by lemma 1.3, $g_{db*db_{n+1}}(w) = \min_{\leq_{f_{db*db_{n+1}}(w)}}(W)$ is a conjunction of literals. But $g_{db*dbn+1}(w) \subseteq val(F)$, so $g_{db*dbn+1}(w) \subseteq val(l_1 \vee \ldots \vee l_m)$. In this case, $\exists k \in \{1 \ldots m\}$ such that $M \models_{MF} B_{db*db_{n+1}} l_k$. This is the same case as before and then $\vdash_{MF} \psi \rightarrow B_{db*db_{n+1}} l_k$. By axiom schemata **A2**, $\vdash_{MF} \psi \rightarrow B_{db*db_{n+1}} l_1 \vee \ldots \vee B_{db*db_{n+1}} l_m$.

(c) if F is a conjunction of clauses, i.e. $F \equiv \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$, then $\models_{MF} \psi \rightarrow B_{db*db_{n+1}} \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$ so $\forall k \in \{1 \ldots n\}$ $\models_{MF} \psi \rightarrow B_{db*db_{n+1}}(l_{k_1} \vee \ldots \vee l_{k_m})$ and this is the same case as previously because of the KD property of $B_{db*db_{n+1}}$.

2. assume that $\models_{MF} \psi \rightarrow \neg B_{db*db_{n+1}} F$.

(a) if F is a literal l of L.

For any MF-model M of ψ and for any world w, $g_{db*db_{n+1}}(w) \not\subset val(l)$. Thus $\exists i \; \exists j \; j \geq i \geq 0$ such that $val(l) \in^i f_{db*db_{n+1}}(w)$ and $val(\neg l) \in^j f_{db*db_{n+1}}(w)$.

By building $f_{db*db_{n+1}}(w)$, we can remark that:

- $\exists i_1 \; \exists i_2$ such that $i_1 + i_2 = i$, $val(l) \in^{i_1} f_{db}$ and $val(l) \in^{i_2} f_{db_{n+1}}$;
- $\exists j_1 \; \exists j_2$ such that $j_1 + j_2 = i$, $val(l) \in^{j_1} f_{db}$ and $val(l) \in^{j_2} f_{db_{n+1}}$;

So:

$$\begin{cases} \models_{MF} \psi \rightarrow B^{i_1}_{db} l & \text{and} \quad \models_{MF} \psi \rightarrow B^{i_2}_{db_{n+1}} l \\ \models_{MF} \psi \rightarrow B^{j_1}_{db} \neg l & \text{and} \quad \models_{MF} \psi \rightarrow B^{j_2}_{db_{n+1}} \neg l \end{cases}$$

$$\text{Lemma 1.4} \quad \Rightarrow \quad \begin{cases} \vdash_{MF} \psi \rightarrow B^{i_1}_{db} l & \text{and} \quad \vdash_{MF} \psi \rightarrow B^{i_2}_{db_{n+1}} l \\ \vdash_{MF} \psi \rightarrow B^{j_1}_{db} \neg l & \text{and} \quad \vdash_{MF} \psi \rightarrow B^{j_2}_{db_{n+1}} \neg l \end{cases}$$

$$\mathbf{A4} \quad \Rightarrow \quad \begin{cases} \vdash_{MF} \psi \rightarrow B^{i}_{db*db_{n+1}} l \\ \vdash_{MF} \psi \rightarrow B^{j}_{db*db_{n+1}} \neg l \end{cases}$$

since $i = i_1 + i_2$ and $j = j_1 + j_2$

There are two cases:

- either $i = j$ and then by axiom **A6**, $\vdash_{MF} \psi \rightarrow \neg B^{i}_{db*db_{n+1}} l$;
- either $j > i$ and then by axiom **A5**, $\vdash_{MF} \psi \rightarrow B^{i}_{db*db_{n+1}} \neg l$. By axiom **A1** $\vdash_{MF} \psi \rightarrow \neg B^{i}_{db*db_{n+1}} l$.

The property is true when F is a literal.

(b) if F is a clause, i.e. $F \equiv l_1 \vee \ldots \vee l_m$, then for any MF-model M of ψ, $M \models_{MF} \neg B_{db*db_{n+1}}(l_1 \vee \ldots \vee l_m)$.

As axiom schemata (A0), (A1) and (A2) are valid, $\models_{MF} B_{db*db_{n+1}}(l_1) \vee \ldots \vee B_{db*db_{n+1}}(l_m) \rightarrow B_{db*db_{n+1}}(l_1 \vee \ldots \vee l_m)$. So $\models_{MF} \neg B_{db*db_{n+1}}(l_1 \vee \ldots \vee l_m) \rightarrow B_{db*db_{n+1}}(l_1) \wedge \ldots \wedge B_{db*db_{n+1}}(l_m)$.

Thus $\forall k \in \{1 \ldots m\}\ M \models_{MF} \neg B_{db*db_{n+1}} l_k$. By the previous proof, as M is a model of ψ, $\forall k \in \{1 \ldots m\} \vdash_{MF} \psi \rightarrow \neg B_{db*db_{n+1}} l_k$. By axiom **A7**, $\vdash_{MF} \psi \rightarrow \neg B_{db*db_{n+1}}(l_1 \vee \ldots \vee l_m)$.

(c) if F is a conjunction of clauses, i.e. $F \equiv \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$, $\models_{MF} \psi \rightarrow \neg B_{db*db_{n+1}} \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$.

Let M be a MF-model of ψ, then $M \models_{MF} \neg B_{db*db_{n+1}} \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \vee \ldots \vee l_{k_m})$, thus $\exists k \in \{1 \ldots m\}\ M \models_{MF} \neg B_{db*db_{n+1}}(l_{k_1} \vee \ldots \vee l_{k_m})$. So:

$$
\begin{array}{lll}
 & \exists k \in \{1 \ldots m\}\ \models_{MF} \psi \rightarrow \neg B_{db*db_{n+1}}(l_{k_1} \vee \ldots \vee l_{k_m}) \\
\Rightarrow & \vdash_{MF} \psi \rightarrow \neg B_{db*db_{n+1}}(l_{k_1} \vee \ldots \vee l_{k_m}) \\
\Rightarrow & \vdash_{MF} \psi \rightarrow \bigvee_{k \in \{1 \ldots m\}} \neg B_{db*db_{n+1}}(l_{k_1} \vee \ldots \vee l_{k_m}) \\
\Rightarrow & \vdash_{MF} \psi \rightarrow \neg(\bigwedge_{k \in \{1 \ldots m\}} B_{db*db_{n+1}}(l_{k_1} \vee \ldots \vee l_{k_m})) \\
\textbf{A1} \text{ and } \textbf{A2} & \vdash_{MF} \psi \rightarrow \neg B_{db*db_{n+1}}(\bigwedge_{k \in \{1 \ldots m\}} (l_{k_1} \vee \ldots \vee l_{k_m}))
\end{array}
$$

Property $\mathcal{H}(n)$ is true for all $n \geq 1$.
□.

Proposition 2. Let ψ be the formula previously defined. Let F be a formula of L and db an information source. Then:

$$\not\vdash_{MF} \psi \rightarrow B_{db}F \iff \vdash_{MF} \psi \rightarrow \neg B_{db}F$$

Proof.
For proving $\Longrightarrow$), we first prove the following lemmas :

Lemma 2.1. Let $M_1 =< W_1, val_1, R_1, B_1 >$ and $M_2 =< W_2, val_2, R_2, B_2 >$ be two MF-models. $\forall w_1 \in W_1\ \exists w_2 \in W_2$ such that $\{l : w_1 \in val(l)\} = \{l : w_2 \in val(l)\}$.

This lemma shows that every world in M_1 has an "equivalent" in M_2.

Lemma 2.2. Let $M_1 =< W_1, val_1, R_1, B_1 >$ and $M_2 =< W_2, val_2, R_2, B_2 >$ be two MF-models. $\forall w_1 \in W_1\ \exists w_2 \in W_2$ such that $\forall F \quad w_1 \in val_1(F) \iff$

$w_2 \in val_2(F)$.

We prove lemma 2.2 by using lemma 2.1 and distinguishing the cases where F is a ground literal, the negation of a formula and the conjunction of two formulas.

Lemma 2.3. Let M_1 and M_2 be two MF-models which satisfy ψ. Then $M_1 \models B_{db}F \iff M_2 \models B_{db}F$.

We prove lemma 2.3 by using lemma 2.1 and 2.2.

Lemma 2.4. Let M be a MF-model. Then $\forall w \in W \forall w' \in W \; g_{db}(w) = g_{db}(w')$.

Now we can prove part $\Longrightarrow$). Let us suppose that $\not\vdash_{MF} \psi \rightarrow B_{db}F$. By proposition 1, $\not\models_{MF} \psi \rightarrow B_{db}l$. Then $\exists M_1$ MF-model of ψ such that $M_1 \not\models B_{db}l$. By lemma 2.3, $\forall M$ MF-model of ψ, $M \not\models B_{db}l$. We prove part $\Longrightarrow$) by distinguishing three cases:

- either F is a literal l and:

$$
\begin{array}{lcll}
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W \quad M, w \not\models B_{db}l & \\
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W \quad \exists w' \in g_{db}(w) \quad w' \notin val(l) & \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W \quad \exists w' \in g_{db}(w) \quad w' \notin val(l) & \text{by lemma 2.4} \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W \quad M, w \not\models B_{db}l & \\
\forall M\ M \models \psi & \Longrightarrow & M \models \neg B_{db}l & \\
 & \Longrightarrow & \models \psi \rightarrow \neg B_{db}l &
\end{array}
$$

- either F is a clause $l_1 \vee \ldots \vee l_n$ and:

$$
\begin{array}{lcll}
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W \quad M, w \not\models B_{db}(l_1 \vee \ldots \vee l_n) & \\
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W \quad \exists w' \in g_{db}(w) \quad w' \not\models l_1 \vee \ldots \vee l_n & \\
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W \quad \exists w' \in g_{db}(w) \quad \forall i \in \{1, \ldots, n\} \quad w' \not\models l_i & \\
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W \quad \exists w' \in g_{db}(w) \quad \forall i \in \{1, \ldots, n\} \quad w' \notin val(l_i) & \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W \quad \forall i \in \{1, \ldots, n\} \quad w \not\models B_{db}(l_i) & \text{by lemma 2.4} \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W \quad M, w \not\models B_{db}(l_1) \vee \ldots \vee B_{db}(l_n) & \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W \quad M, w \models \neg(B_{db}(l_1) \vee \ldots \vee B_{db}(l_n)) & \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W \quad M, w \models \neg B_{db}(l_1 \vee \ldots \vee l_n) & \text{as axiom} \\
 & & \text{schemata (A7) is sound} & \\
 & \Longrightarrow & \models \psi \rightarrow \neg B_{db}(l_1 \vee \ldots \vee l_n) &
\end{array}
$$

- either F is a conjunction of clauses $C_1 \wedge \ldots \wedge C_m$ and:

$$\begin{array}{lll}
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W\ \ M, w \not\models B_{db}(C_1 \wedge \ldots \wedge C_m) \\
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W\ \ \exists w' \in g_{db}(w)\ \ w' \not\models C_1 \wedge \ldots \wedge C_m \\
\forall M\ M \models \psi & \Longrightarrow & \exists w \in W\ \ \exists w' \in g_{db}(w)\ \ \exists i \in \{1, \ldots, m\}\ \ w' \not\models C_i \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W\ \ \exists i \in \{1, \ldots, n\}\ \ w \not\models B_{db}(C_i) \text{ by lemma 2.4} \\
\forall M\ M \models \psi & \Longrightarrow & \forall w \in W\ \ w \not\models B_{db}(C_1) \wedge \ldots \wedge B_{db}(C_m) \\
\forall M\ M \models \psi & \Longrightarrow & M \models \neg(B_{db}(C_1) \wedge \ldots \wedge B_{db}(C_m)) \\
\forall M\ M \models \psi & \Longrightarrow & M \models \neg B_{db}(C_1 \wedge \ldots \wedge C_m) \text{ as axiom schematas} \\
 & & \text{(A0), (A1) and (A2) are sound} \\
 & \Longrightarrow & \models \psi \rightarrow \neg B_{db}(C_1 \wedge \ldots \wedge C_m)
\end{array}$$

For proving $\Longleftarrow$), let us assume that $\vdash_{MF} \psi \rightarrow B_{db}F$ and $\vdash_{MF} \psi \rightarrow \neg B_{db}F$. In this case, by proposition 1, $\models_{MF} \psi \rightarrow B_{db}F$ and $\models_{MF} \psi \rightarrow \neg B_{db}F$. Thus, for any MF-model M of ψ, $M \models_{MF} B_{db}F$ and $M \models_{MF} \neg B_{db}F$. This is impossible (because ψ has at least one model), so $\vdash_{MF} \psi \rightarrow \neg B_{db}F \implies \not\vdash_{MF} \psi \rightarrow B_{db}F$.

□.

Proposition 3. Let $db_1 ... db_n$ be n consistent sets of literals to be merged and F be a formula of L. With the notations introduced in section 2 and 3, we have:

$$\vdash \psi \rightarrow B_{(\ldots(db_1 * db_2) * \ldots db_n)}F \Longleftrightarrow \Delta_\Sigma([db_1 ... db_n]) \models F$$

Proof.

We first notice that if $M =< W, val, R, B >$ is a MF-model, then for any world w in W, there is an interpretation w' of L such that $\{l : w' \models l\} = \{l : w \in val(l)\}$ (see lemmas 2.1 and 2.2). And if w' is an interpretation of L, then there is a world w in W such that $\{l : w' \models l\} = \{l : w \in val(l)\}$. In other terms, any world in W correspond to an interpretation of L and any interpretation of L corresponds to (at least) one world in W.

Assume now that $\forall i = 1...n\ \ db_i = \{l_i^1 ... l_i^{m_i}\}$.

Let $M =< W, val, R, B >$ is a MF-model which satisfies ψ. Let w be a world in W and w' the interpretation of L previously characterized. We prove that:

$$d_{sum}(w, [...val(l_1^i)...val(l_n^j)...]) = dist_\Sigma(w', [db_1 ... db_n])$$

This shows that the W worlds which are minimal according to the order induced by the distance d_{sum} correspond to the interpretations of L which are minimal according to the order induced by the distance d_Σ.

□.

Proposition 4. Let l be a literal, let db denoting an information source (primitive or not). Then, using negation-as-failure on the meta-program META,

(1) PROLOG succeeds in proving $B(db, l)$ iff $\models_{MF} (\psi \rightarrow B_{db} l)$
(2) PROLOG fails in proving $B(db, l)$ iff $\models_{MF} (\psi \rightarrow \neg B_{db}\ l)$

Proof.
For proving (1), we first prove that PROLOG succeeds in proving $R(db, l, i)$ iff $\models_{MF} \psi \rightarrow B^i_{db} l$. We prove this lemma by induction on the length of PROLOG's proof. Proof of (1) is then obvious.
(2) derives from (1) and from proposition 2.
□.

Proposition 5. Let $DB =< EDB, IDB >$ a database which is equivalent to a set of ground literals. Let $l_1 ... l_n$ be some ground literals of LO such that $l_1 \vee ... \vee l_n$ is not a tautology. Then,
$EDB \cup IDB \models l_1 \ \vee \ ... \ \vee \ l_n$ iff $\exists i_0 \in \{1...n\}\ \ EDB \cup IDB \models l_{i_0}$

Proof
The proof of $\Longleftarrow$) is obvious.

Proof of $\Longrightarrow$)
Let $l_1 ... l_n$ be some ground literals of LO such that $l_1 \vee ... \vee l_n$ is not a tautology and $EDB \cup IDB \models l_1 \ \vee \ ... \ \vee \ l_n$.
Let us assume that $\forall i \in \{1, \ldots, n\}\ EDB \cup IDB \not\models l_i$ (hyp).
On the one hand, let us denote $\{i_1, \ldots, i_m\}$ the minimal subset of $\{1, \ldots, n\}$ such that $EDB \cup IDB \models l_{i_1} \vee \ldots \vee l_{i_m}$ (i.e. $\forall j \in (\{1, \ldots, n\} - \{i_1, \ldots, i_m\})\ EDB \cup IDB \models \neg l_j$). Since $EDB \cup IDB \models l_{i_1} \vee \ldots \vee l_{i_m}\ \forall i \in \{i_1, \ldots, i_m\}\ \exists HM_i$ Herbrand model of $EDB \cup IDB$ such that $HM_i \models l_i$.
Moreover, notice that $m > 1$, else $\forall HM$ Herbrand model of $EDB \cup IDB\ \ HM \models l_{i_1}$ which contradicts (hyp).
On the other hand, $\forall i \in \{i_1, \ldots, i_m\}\ \exists HM_i$ Herbrand model of $EDB \cup IDB$ such that $HM_i \models \neg l_i$, because of (hyp).
Thus, $\forall i \in \{i_1, \ldots, i_m\}\ l_i \in L$ and $\neg l_i \in L$ (cf. definition 10).
$l_{i_1} \vee \ldots \vee l_{i_m}$ is not a tautology, so $\forall i \in \{i_1, \ldots, i_m\}\ \forall j \in (\{i_1, \ldots, i_m\} - \{i\})$ $l_i \not\equiv \neg l_j$. Thus, $\neg l_{i_1} \wedge \ldots \wedge \neg l_{i_m}$ is satisfiable.
In this case, by definition 10, as $< EDB, IDB >$ is equivalent to a set of ground literals, $\exists HM_{i_0}$ Herbrand model of $EDB \cup IDB$ such that $HM_{i_0} \models \neg l_{i_1} \wedge \ldots \neg \wedge l_{i_m}$. Thus, by 1., $HM_{i_0} \models \neg l_1 \wedge \ldots \neg \wedge l_n$.
This contradicts the fact that $EDB \cup IDB \models l_1 \vee \ldots \vee l_n$, so (hyp) is false.
□.

Proposition 6. Let $db =< EDB, IDB >$ such that IDB is not recursive. Let l be a ground literal. Then PROLOG proves $B_{exp}(db, l)$ iff $EDB \cup IDB \models l$.

Proof.
Proof of $\Longrightarrow$)

We prove in the same time the two following propositions:
(1) PROLOG proves $B_{exp}(db, l) \Longrightarrow EDB \cup IDB \models l$ and
(2) PROLOG proves $B_{conj}(db, l_1 \wedge \ldots \wedge l_n) \Longrightarrow EDB \cup IDB \models l_1 \wedge \ldots \wedge l_m$

Let us denote:

$\mathcal{H}(n)$: PROLOG proves $B_{exp}(db, l)$ in n steps $\Longrightarrow EDB \cup IDB \models l$
PROLOG proves $B_{conj}(db, l_1 \wedge \ldots \wedge l_m)$ in $n+1$ steps $\Longrightarrow$
$EDB \cup IDB \models l_1 \wedge \ldots \wedge l_m$

We will prove part $\Longrightarrow$) of (1) and (2) by proving by induction that $\mathcal{H}(n)$ is true for any $n \geq 2$.

First, let us prove that PROLOG will always prove $B_{exp}(db, l)$ or $B_{conj}(db, l_1 \wedge \ldots \wedge l_m)$ in a finite number of steps.

Let us suppose that PROLOG cannot prove $B_{exp}(db, l)$ or $B_{conj}(db, l_1 \wedge \ldots \wedge l_m)$ in a finite number of steps. In this case, it means that PROLOG tries to prove some formula $B_{exp}(db, l')$ or $B_{conj}(db, l'_1 \wedge \ldots \wedge l'_m)$ by using the same formula (look at axioms (1.4) and (1.6)). These cases can only happen when PROLOG can prove $IDB(db, r \rightarrow l')$ where l' appears in the conjunction r. But IDB is not recursive, so this case can never happen.

Proof of $\mathcal{H}(2)$: PROLOG proves $B_{exp}(db, l)$ in two steps. In this case, the first axiom used by PROLOG is (1.3). PROLOG has then to prove $EDB(db, l)$ using axiom (1.1). Proving $EDB(db, l)$ means that $l \in EDB$. Thus $EDB \cup IDB \models l$.

PROLOG proves $B_{conj}(db, l_1 \wedge \ldots \wedge l_m)$ in three steps. The only axiom PROLOG can use for first step is (1.6). So PROLOG has to prove:

- $B_{conj}(db, l_2 \wedge \ldots \wedge l_m)$ in two steps. The only axiom PROLOG can use is (1.5), so in fact $l_2 \wedge \ldots \wedge l_m \equiv nil$.
- $B_{exp}(db, l)$ in two step. By the previous proof, $EDB \cup IDB \models l_1$.

Thus, $EDB \cup IDB \models l_1 \wedge \ldots \wedge l_m$ and $\mathcal{H}(2)$ is true.

Let us prove also that $\mathcal{H}(3)$ is true because it is a particular case.

Proof of $\mathcal{H}(3)$: let us suppose that PROLOG proves $B_{exp}(db, l)$ in three steps. By axiom (1.4), PROLOG has then to prove $IDB(db, r \rightarrow l)$ and $B_{conj}(db, r)$ in two steps. But PROLOG cannot prove $B_{conj}(db, r)$ in two steps (it needs at least three steps to prove a B_{conj} formula), so the proposition "PROLOG proves $B_{exp}(db, l)$ in three steps $\Longrightarrow EDB \cup IDB \models l$" is true.

PROLOG proves $B_{conj}(db, l_1 \wedge \ldots \wedge l_m)$ in four steps. The only axiom PROLOG can use for first step is (1.6). So PROLOG has to prove:

- $B_{conj}(db, l_2 \wedge \ldots \wedge l_m)$ in three steps. As $\mathcal{H}(2)$ is true, $EDB \cup IDB \models l_2 \wedge \ldots \wedge l_m$.

- $B_{exp}(db, l)$ in at most three steps. So PROLOG proves $B_{exp}(db, l)$ in two steps. As $\mathcal{H}(2)$ is true, $EDB \cup IDB \models l_1$.

Thus, $EDB \cup IDB \models l_1 \wedge \ldots \wedge l_m$ and $\mathcal{H}(3)$ is true.

Induction step for $\mathcal{H}$: let n be an integer such that $n \geq 3$. Let us suppose that $\mathcal{H}(k)$ is true for any $k \in \{3, \ldots, n\}$.

First, let us suppose that PROLOG proves $B_{exp}(db, l)$ in $(n+1)$ steps. In this case, by axiom (1.4) and (1.2):

- PROLOG proves $B_{conj}(db, r)$ in n steps. As $\mathcal{H}(n-1)$ is true, $EDB \cup IDB \models r$.
- PROLOG proves $IDB(db, r \rightarrow l)$ in two steps. So $EDB \cup IDB \models r \rightarrow l$.

Thus $EDB \cup IDB \models l$.

Let us suppose that PROLOG proves $B_{conj}(db, l_1 \wedge \ldots \wedge l_m)$ in $n+2$ steps. In this case, by axiom (1.6):

- PROLOG proves $B_{exp}(db, l_1)$ in $n+1$ steps. By the previous proof, $EDB \cup IDB \models l_1$.
- PROLOG proves $B_{conj}(db, l_2 \wedge \ldots \wedge l_m)$ in $n+1$ steps. As $\mathcal{H}(n)$ is true, $EDB \cup IDB \models l_2 \wedge \ldots \wedge l_m$.

Thus $EDB \cup IDB \models l_1 \wedge \ldots \wedge l_m$, thus $\mathcal{H}(n+1)$ is true.

So $(H)(n)$ is true for any $n \geq 2$. So PROLOG proves $B_{exp}(db, l) \Longrightarrow EDB \cup IDB \models l$.

Proof of $\Longleftarrow$)

Let us assume that LO is a first order language such that the constants symbols of LO are denoted $\{a_1, \ldots, a_n\}$, the predicates symbols $\{P_1, \ldots, P_m\}$ and the variables symbols $\{x_1, \ldots, x_k\}$. For any clause $\mathcal{C}$ of IDB containing the variables $x_{i_1}, \ldots, x_{i_{k_\mathcal{C}}}$, we will call a ground instance of $\mathcal{C}$ any formula $\mathcal{F}$ of LO such that $\mathcal{F} \equiv \mathcal{C}[x_{i_1}/a_{j_1}, \ldots, x_{i_{k_\mathcal{C}}}/a_{j_{k_\mathcal{C}}}]$ (i.e. a clause in which each variable symbol is replaced by a constant symbol).

Let us define $succ$ the fonction such that for any set E of ground literals of LO:

$$\begin{aligned} succ(E) \quad = \quad & E \cup \{P_i(a_{i_1}, \ldots, a_{i_{m_i}}) \ : \ \neg P_{j_1}(a_{j_{1,1}}, \ldots, a_{j_{1,l_1}}) \wedge \ldots \wedge \\ & \neg P_{j_h}(a_{j_{h,1}}, \ldots, a_{j_{h,l_h}}) \rightarrow P_i(a_{i_1}, \ldots, a_{i_{m_i}}) \text{ is a ground instance} \\ & \text{of a clause of } h(c) \text{ where } c \in IDB \text{ and} \\ & \forall k \in \{j_1, \ldots, j_h\} \ \ \neg P_{j_k}(a_{j_{1,1}}, \ldots, a_{j_{1,l_k}}) \in E\} \end{aligned}$$

Let us define fonction $\mathcal{T}$ by induction :

$$\begin{aligned} \mathcal{T}_1(EDB) \quad &= \quad EDB \\ \forall n \geq 1 \ \ \mathcal{T}_{n+1}(EDB) \quad &= \quad succ(\mathcal{T}_n(EDB)) \end{aligned}$$

As LO is a finite language, it is easy to prove that $\exists i_0 \geq 1$ such that $\lim_{n\to\infty} \mathcal{T}_n(EDB) = \mathcal{T}_{i_0}(EDB)$ ($succ$ is a growing series in a finite set). It is also obvious that for any ground literal l, $EDB \cup IDB \models l \Longleftrightarrow l \in \mathcal{T}_{i_0}(EDB)$ Let us denote $\mathcal{T}_{i_0}(EDB)$ as $\mathcal{T}^{io}(EDB)$.

Let l be some ground literal. Let us prove by induction that for any $i \geq 1$:

$$\mathcal{K}(i): \quad l \in \mathcal{T}^i(EDB) \Longrightarrow \text{PROLOG proves } B_{exp}(db, l).$$

Proof of $\mathcal{K}(1)$: $l \in \mathcal{T}^1(EDB)$, so by definition $l \in EDB$. By axioms (1.1) and (1.3), PROLOG proves $B_{exp}(db, l)$.

Induction step for $\mathcal{K}$: let us suppose that for some $i \geq 1$ $\mathcal{K}(i)$ is true. Let us suppose that $l \in \mathcal{T}^{i+1}(EDB)$. In this case, there is a formula $\neg P_{j_1}(a_{j_{1,1}}, \ldots, a_{j_{1,l_1}}) \wedge \ldots \wedge \neg P_{j_h}(a_{j_{h,1}}, \ldots, a_{j_{h,l_h}}) \rightarrow l$ which is an instance of a clause in IDB such that $\forall k \in \{j_1, \ldots, j_h\}$ $\neg P_{j_k}(a_{j_{1,1}}, \ldots, a_{j_{1,l_k}}) \in \mathcal{T}_i(EDB)$.

By axiom (1.2), PROLOG proves $IDB(db, \neg P_{j_1}(a_{j_{1,1}}, \ldots, a_{j_{1,l_1}}) \wedge \ldots \wedge \neg P_{j_h}(a_{j_{h,1}}, \ldots, a_{j_{h,l_h}}) \rightarrow l)$.

As $\forall k \in \{j_1, \ldots, j_h\}$ $\neg P_{j_k}(a_{j_{1,1}}, \ldots, a_{j_{1,l_k}}) \in \mathcal{T}_i(EDB)$ and $\mathcal{K}(i)$ is true, PROLOG proves $B_{exp}(db, \neg P_{j_k}(a_{j_{1,1}}, \ldots, a_{j_{1,l_k}}))$ for any $k \in \{j_1, \ldots, j_h\}$. So, by axiom (1.5) and (1.6), PROLOG proves $B_{conj}(db, \neg P_{j_1}(a_{j_{1,1}}, \ldots, a_{j_{1,l_1}}) \wedge \ldots \wedge \neg P_{j_h}(a_{j_{h,1}}, \ldots, a_{j_{h,l_h}}))$.

Thus, by axiom (1.4), PROLOG proves $B_{exp}(db, l)$. So $\mathcal{K}(i+1)$ is true.

By induction, $\mathcal{K}(i)$ is true for any $i \geq 1$. In particular, $\mathcal{K}(i_0)$ is true, so PROLOG proves $EDB \cup IDB \models l \Longrightarrow B_{exp}(db, l)$.

□.