Answering queries addressed 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 addressed 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 addressed 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], [Suh94], [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 $M_{S_1} = [S_1, ..., S_n]$ and $M_{S_2} = [S_{n+1}, ..., S_m]$ be two multi-sets. The union of two multi-sets is defined by: $M_{S_1} \cup M_{S_2} = [S_1, ..., S_m]$. The membership relation is defined by: $S \in^i M S$ iff there are exactly $i$ occurrences of $S$ in the multi-set $M S$. Notice that, in the limit case, $S \in^0 M S$ iff there is no occurrence of $S$ in $M S$, i.e. $S \notin M S$.

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 $db'$. 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

\footnote{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: $B_{db}^i$ and $B_{db}$, where $i$ is an integer and $db$ denotes an information source (primitive or not).

We expect that the formula $B_{db}^i l$ 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 $B_{db}^i$ 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 $B_{db}^i$ and $B_{db}$ are modal operators, then $B_{db}^i F$ and $B_{db} F$ are formulas of $L'$. If $F_1$ and $F_2$ are formulas of $L'$, then, $\neg F_1$, $F_1 \land F_2$, $F_1 \lor F_2$ and $F_1 \rightarrow F_2$ are defined from the previous ones as usual.

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: $B_{db_1}^0$, $B_{db_1}$, $B_{db_2}^0$, $B_{db_2}$, $B_{db_1}^1 \star db_1$, $B_{db_1}^1 \star db_2$, $B_{db_2}^1 \star db_1$, and $B_{db_2}^1 \star db_2$.

We expect that, for instance: $B_{db_1}^1 a$ means that $db_1$ contains one occurrence of $a$. $B_{db_2}^0 a$ means that $db_2$ contains no occurrence of $a$. $B_{db_1}^1 \star db_2 a$ means that the information source obtained by merging $db_1$ and $db_2$ contains one occurrence of $a$. Finally, $B_{db_1} \star db_2 f a$ 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, $val(\neg P) = W \setminus val(P)$, $val(P \land Q) = val(P) \cap val(Q)$. 

4
• $B$ 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:
\[ \forall w \in W \ f_{db+db'}(w) = f_{db}(w) \cup f_{db'}(w) \]

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

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 $\leq_{fa}(w)$. 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).

\[
M, w \models_M F \text{ iff } w \in val(p)
\]

\[
M, w \models_M \neg F_1 \text{ iff } M, w \not\models_M F_1
\]

\[
M, w \models_M F_1 \land F_2 \text{ iff } M, w \models_M F_1 \text{ and } M, w \models_M F_2
\]

\[
M, w \models_M B^{db}_i F \text{ iff } \forall \upsilon (F) \in i f_{db}(w)
\]

\[
M, w \models_M g_{db}(w) \subseteq val(F)
\]

**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 $MF$, $\forall w \in W$, $M, w \models_M F$. We note $\models_M F$. 

5
2.4 Proof theory

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

The axiom schemata of $MF$ are:

$(A_0)$ Axiom schemata of propositional logic

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

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

$(A_3)$ $B_{db}^i l \rightarrow \neg B_{db}^j l$ if $i \neq j$

$(A_4)$ $B_{db}^k l \land B_{db}^i l \rightarrow B_{db}^{k+i} l$ if $k = i + j$

$(A_5)$ $B_{db}^i l \land B_{db}^j \neg l \rightarrow B_{db}^i l$ if $i < j$

$(A_6)$ $B_{db}^i l \land B_{db}^j \neg l \rightarrow \neg B_{db}^i l$

$(A_7)$ $B_{db} (l_1 \lor \ldots \lor l_n) \rightarrow B_{db} l_1 \lor \ldots \lor B_{db} l_n$ with $\forall i \in \{1 \ldots n\} \forall j \in \{1 \ldots n\}$ $l_i \neq \neg l_j$

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, $(A_0), (A_1), (A_3)$.

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

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

$(A_5), (A_6)$ 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 than 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.

$(A_7)$ 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 $a \lor \neg a$ is believed (due to Nec) even though neither $a$ nor $\neg a$ is believed. $(A_7)$ prevents the case when a database believes, for instance $a \lor b$ 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 $\psi$ by:

$$\psi = \bigwedge_{i=1}^{n} \left( \bigwedge_{l \in db_i} B_{db_i}^l \land \bigwedge_{l \in \bar{db}_i} B_{\bar{db}_i}^l \right)$$

$\psi$ 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 $\psi$ 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:

$$\models_{MF} \psi \rightarrow B_{db}F \iff \vdash_{MF} \psi \rightarrow B_{db}F \text{ and}$$

$$\models_{MF} \psi \rightarrow \neg B_{db}F \iff \vdash_{MF} \psi \rightarrow \neg B_{db}F$$

The proof is given in the appendix.

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

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

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: $db_1 = \{a, b\}$, $db_2 = \{a, \neg c\}$, $db_3 = \{-a, c\}$. By definition 7, $\psi$ is: $B_{db_1}^1 a \land B_{db_2}^1 b \land B_{db_2}^0 c \land B_{db_3}^0 \neg c \land B_{db_3}^0 \neg a \land B_{db_3}^0 \neg b \land B_{db_2}^0 \neg a \land B_{db_2}^0 \neg c \land B_{db_2}^0 \neg \neg b \land B_{db_2}^0 \neg a \land B_{db_2}^0 \neg c \land B_{db_2}^0 \neg \neg a \land B_{db_2}^0 \neg c$.

Here are some theorems of $MF$ we can derive:

$(\alpha) \vdash \psi \rightarrow B_{db_2}^1 a$ (by $(A1)$)

$(\beta) \vdash \psi \rightarrow B_{db_2}^0 a$ (by $(\alpha)$ and $(A4)$)

$(\gamma) \vdash \psi \rightarrow B_{db_3}^0 a$ (by $(A1)$)

$(\delta) \vdash \psi \rightarrow B_{db_2}^1 a$ (by $(\gamma)$ and $(A4)$)

Thus, finally, from $(\beta)$, $(\delta)$ and $(A5)$, we can prove:

$(\zeta) \vdash \psi \rightarrow B_{db_2}^1 a$

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:
\((\eta) \vdash \psi \rightarrow B_{(d_{b_1} \ast d_{b_2}) \ast d_{b_3}} b\)

Thus, from \((\zeta), (\eta), (A_0)\) and \((A_2)\) we prove:

\(\vdash \psi \rightarrow B_{(d_{b_1} \ast d_{b_2}) \ast d_{b_3}} (a \land b)\)

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

Similarly, we have:

\(\theta : \vdash MF \psi \rightarrow B^{d_{b_1} \ast d_{b_2} c} (by \ (A_4))\)

\(t : \vdash MF \psi \rightarrow B^{1_{d_{b_1} \ast d_{b_2} \ast d_{b_3}}} c (by \ (\theta) \ and \ (A_4))\)

\(\kappa : \vdash MF \psi \rightarrow B^{1_{d_{b_1} \ast d_{b_2} \lnot c} (by \ (A_4))}\)

\(\lambda : \vdash MF \psi \rightarrow B^{1_{d_{b_1} \ast d_{b_2} \ast d_{b_3} \lnot c} (by \ (\theta) \ and \ (A_4))}\)

\(\mu : \vdash MF \psi \rightarrow \lnot B^{1_{d_{b_1} \ast d_{b_2} \ast d_{b_3} \lnot c} (by \ (\theta) \ and \ (A_4))}\)

\(\nu : \vdash MF \psi \rightarrow \lnot B_{(d_{b_1} \ast d_{b_2}) \ast d_{b_3} \lnot c} (by \ (\theta) \ lnot \ c)\)

Thus, finally, by \((\mu), (\nu)\) and \((A_0)\) we can prove:

\(\vdash MF \psi \rightarrow \lnot B_{(d_{b_1} \ast d_{b_2}) \ast d_{b_3} \lnot c} (by \ (\theta) \ lnot \ c)\)

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

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 \(db_1, \ldots, db_n\) be n information sources to be merged.

Konieczny and Pino-Pérez define a majority merging operator, denoted \(\Delta_{\Sigma}\), such that the models of the information source which is obtained from merging \(db_1 \ldots db_n\) with this operator, is semantically characterized by:

\[ Mod(\Delta_{\Sigma}([db_1, \ldots, db_n])) = \min_{\leq_{\Sigma} [db_1, \ldots, db_n]} \text{(W)} \]

where W denotes the set of all the interpretations of the language L (the propositional language used to describe the contents of the information sources).

\(\leq_{\Sigma} [db_1, \ldots, db_n]\) is a total pre-order on W defined by:

\[ w \leq_{\Sigma} [db_1, \ldots, db_n] u' \iff d_{\Sigma}(w, [db_1 \ldots db_n]) \leq d_{\Sigma}(w', [db_1 \ldots db_n]) \]

with

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

\(^3\)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 $\Delta_{\Sigma}$, the result is semantically characterized by the interpretations which are minimal according to the pre-order $\preceq^\Delta_{\Sigma}[db_1,...,db_n]$. 

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

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

*The proof is given in the appendix.

In other words, the information source whose beliefs are characterized by theorems $\vdash \psi \rightarrow B_{[(\ldots(db_1*db_2)*\ldots db_n) F]}$ 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: $\psi \rightarrow B_{\Delta}F$. Extension to non-atomic formulas will be discussed in section 6.

One will notice that in this prover, the formula $\psi$ introduced previously will not be used. Indeed, $\psi$ 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 information sources plus a constant symbol denoted $\text{nil}$ and constants denoting integers: $1$, $2$, etc.

- a binary function noted $\ast$. By convention, $(db_{i_1} \ast \ldots \ast db_{i_k})$ represents the term: $db_{i_1} \ast (db_{i_2} \ldots \ast (db_{i_k} \ast \text{nil}) \ldots)$. This function will be used to denote the information sources obtained by merging information sources $db_{i_1}...db_{i_k}$. 

9
• a binary function denoted + which is the sum of integers.

• a unary function symbol ¬. By convention, ¬I represents the term ¬(I).
  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) \( ¬NIL(db_2) \lor R(db_1, l, i) \lor R(db_2, l, j) \land (k = i + j) \rightarrow R(db_1 * db_2, l, k) \)
(3) \( NIL(db_2) \lor B_{\exp}(db_1, l) \rightarrow R(db_1 * db_2, l, 1) \)
(4) \( NIL(db_2) \lor ¬B_{\exp}(db_1, l) \rightarrow R(db_1 * db_2, l, 0) \)
(5) \( R(db, l, i) \lor R(db, ¬l, j) \land (i > j) \rightarrow B(db, l) \)
(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 \( \models_{MF} (\psi \rightarrow ¬B_{db} l) \)
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 $DB = < EDB, IDB >$ 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 $IDB_1 = \\{\neg p(x) \land \neg q(x), p(x) \land r(x)\}$. The Herbrand models of $DB_1$ are $\{p(a)\}$ 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 $IDB_2 = \\{\neg p(x) \land q(x) \land r(x)\}$. 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 $\neg q(a) \land \neg r(a)$. 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 $l_1 \lor...\lor l_n$ is not a tautology. Then,
\[ EDB \cup IDB \models l_1 \lor \ldots \lor l_n \text{ iff } \exists i_0 \in \{1\ldots n\} \ EDB \cup IDB \models l_{i_0} \]

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:

\[ h(l_1 \lor \ldots \lor l_n) = \{ (\neg l_1 \land \ldots \land \neg l_{i-1} \land \neg l_{i+1} \land \ldots \land \neg l_n) \rightarrow l_i, \ i \in \{1\ldots n\} \} \]

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

1. \( EDB(db, l) \) if the ground literal \( l \) is in the \( EDB \) part of the database \( db \).
2. \( IDB(db, f) \) if the formula \( f \) is in \( h(c) \), where \( c \) is a clause in the \( IDB \) part of \( db \).
3. \( EDB(db, l) \rightarrow B_{exp}(db, l) \)
4. \( IDB(db, (r \rightarrow l)) \land B_{conj}(db, r) \rightarrow B_{exp}(db, l) \)
5. \( B_{conj}(db, nil) \)
6. \( B_{exp}(db, l_1) \land B_{conj}(db, r_1) \rightarrow B_{conj}(db, l_1 \land r_1) \)

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:

$$answer((db_1 \ldots \ast db_n), F) = \text{YES} \iff \text{PROLOG proves } B((db_1 \ldots \ast db_n), F)$$

$$answer((db_1 \ldots \ast db_n), F) = \text{NO} \iff \text{PROLOG proves } B((db_1 \ldots \ast db_n), \neg F)$$

$$answer((db_1 \ldots \ast db_n), F) = ? \text{ else}$$

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 $B((db_1 \ldots \ast db_n), F)$ and $B((db_1 \ldots \ast db_n), \neg F)$ 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:

$$answer((db_1 \ldots \ast \ldots \ast db_n), F(X)) = \{A : \text{tuple of constant symbols such that PROLOG proves } B((db_1 \ldots \ast \ast db_n), F(A))\}$$

5.3 Example

Let us consider the three following databases:

$$db_1 = \langle EDB_1, IDB \rangle, db_2 = \langle EDB_2, IDB \rangle, db_3 = \langle EDB_3, IDB \rangle$$

with:

$$EDB_1 = \{\text{student(John)}, \text{employee(Louis)}, \text{self(Philip)}, \text{self(Donald)}, \text{disabled(Louis)}\}$$

$$EDB_2 = \{\text{employee(Philip)}, \text{student(Louis)}, \text{restaurant(John)}, \text{restaurant(Henry)}\}$$

$$EDB_3 = \{\text{student(John)}, \text{employee(Philip)}\}$$

$$IDB = \{\forall x \text{ student}(x) \rightarrow \text{self}(x), \forall x \text{ employee}(x) \rightarrow \text{restaurant}(x),$$

$$\forall x \sim \text{self}(x) \vee \sim \text{restaurant}(x), \forall x \text{ disabled}(x) \rightarrow \text{parking}(x),$$

$$\forall x \text{ employee}(x) \rightarrow \text{parking}(x), \forall x \text{ student}(x) \rightarrow \sim \text{parking}(x)\}$$

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 $db_3$ ?

$\text{answer}((db_1 * db_2 * db_3), \text{student}(John)) = YES$

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 $db_3$ ?

$\text{answer}((db_1 * db_2 * db_3), \neg \text{employee}(John)) = YES$

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 $db_3$ ?

$\text{answer}((db_1 * db_2 * db_3), \text{student}(Donald)) = ?$

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 $db_2$ ?

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

When considering the first two databases, there is no student.

Who is student in the database obtained by merging $db_1, db_2$ and $db_3$ ?

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

When considering the three databases, John is the only student.

Who is employee in the database obtained by merging $db_1, db_2$ and $db_3$ ?

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

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 $db_3$ ?

$\text{answer}((db_1 * db_2 * db_3), \text{parking}(x)) = \{Philip\}$

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 "strength" 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($(db_1 \ast db_2 \ast db_3), \text{self}(x)) = \{\text{John}, \text{Donald}\}$

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($(db_1 \ast db_2 \ast db_3), \text{restaurant}(x)) = \{\text{Philip}, \text{Henry}\}$

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($(db_1 \ast db_2), \text{student}(\text{John})) = ?$

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 interpreter 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 $\land$ and $\lor$ and the evaluator is extended by adding the following meta-axioms:

$C(db, \text{nil})$
$D(db, s) \land C(db, c) \rightarrow C(db, d \land c)$
$B(db, l) \rightarrow D(db, l \lor d)$
$B(db, d) \rightarrow D(db, l \lor d)$

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 $db_1 \ldots db_n$?” is YES, NO or ? depending on the fact that PROLOG proves $C((db_1 \ast \ldots \ast db_n, F)$, $C((db_1 \ast \ldots \ast db_n, \neg F)$, 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 \(C(db_1 \ast ... \ast db_n, F(A))\).

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 \((db_1 \ast ... \ast db_n) ?\)” 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 \((db_1 \ast ... \ast db_n) ?\)” when there is a no database which believes Q and also when there are as many databases which believe Q and databases which believe \(\neg Q\).

If one wants to distinguish these different kinds of situations, we suggest to replace the modalities \(B_{db}\) by four new modalities: \(\text{Unchallenged}_{db}, \text{Majority}_{db}, \text{CompleteLack}_{db}\) and \(B1_{db}\). \(\text{Unchallenged}_{db}\) intends to mean that in \(db\), there are proofs for \(l\) but no proof against \(l\); \(\text{Majority}_{db}\) intends to mean that \(db\) believes \(l\) by majority; \(\text{CompleteLack}_{db}\) intends to mean that in \(db\), there is a complete lack of information about \(l\) and \(B1_{db}\) 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:

\[(A5.1) B^i_{db} \land B^j_{\neg l} \rightarrow \text{Unchallenged}_{db} \text{ for } i > 0 \]
\[(A5.2) B^i_{db} \land B^{j \neq i}_{\neg l} \rightarrow \text{Majority}_{db} \text{ for } i > j > 0 \]
\[(A6.1) B^i_{db} \land B^i_{\neg l} \rightarrow \text{CompleteLack}_{db} \]
\[(A6.2) B^i_{db} \land B^j_{\neg l} \rightarrow \text{BI}_{db} \text{ for any } i > 0 \]

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:

\[(5.1) R(db, l, 0) \land R(db, \neg l, 0) \rightarrow B(\text{CompleteLack}, db, l) \]
\[(5.2) R(db, l, i) \land R(db, l, 0) \land (i > 0) \rightarrow B(\text{Unchallenged}, db, l) \]
\[(5.3) R(db, l, i) \land R(db, \neg l, i) \land (i > 0) \rightarrow B(\text{Majority}, db, l) \]
\[(5.4) R(db, l, i) \land R(db, \neg l, j) \land (i > j) \land (j > 0) \rightarrow B(\text{BI}, db, l) \]

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:

\[\text{answer}((db_1 \ast db_2 \ast db_3), \text{student}(John)) = \text{YES (by majority)} \]
\[\text{answer}((db_1 \ast db_2 \ast db_3), \neg \text{employee}(John)) = \text{YES (there is no proof against it)} \]
\[\text{answer}((db_1 \ast db_2 \ast db_3), \text{student}(Donald)) = \text{Don’t know (complete lack of information)} \]
\[\text{answer}((db_1 \ast db_2), \text{student}(John)) = \emptyset \]
\[\text{answer}((db_1 \ast db_2 \ast db_3), \text{student}(x)) = \{\text{John (by majority)}\} \]
\[\text{answer}((db_1 \ast db_2 \ast db_3), \text{employee}(x)) = \{\text{Philip (by majority)}\} \]
\[\text{answer}((db_1 \ast db_2 \ast db_3), \text{parking}(x)) = \{\text{Philip (there is no proof against it)}\} \]
answer((db1 ∗ db2 ∗ db3), self(x)) = {John (by majority), Donald (there is no proof against it)}
answer((db1 ∗ db2 ∗ db3), restaurant(x)) = {Philip (by majority), Henry (by majority)}
answer((db1 ∗ db2), 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


Appendix: proofs

**Proposition 1.** Let $\psi$ be the formula defined by definition 7. Let $F$ be a formula of $L$ and $db$ an information source. Then we have:

$\vdash MF \psi \rightarrow B_{db} F \iff \vdash MF \psi \rightarrow B_{db} F$ and

$\vdash MF \psi \rightarrow \neg B_{db} F \iff \vdash MF \psi \rightarrow \neg B_{db} F$

**Proof.**

$\iff$). 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.

$\implies$). We first prove the four following lemmas:

**Lemma 1.1.** Let $M = < W, val, R, B >$ be a model in which $\psi$ is satisfied. Then, for any primitive information source $db$, we have: $f_{db} = \{ val(l) : l \in db \}$

**Lemma 1.2.** Let $M = < W, val, R, B >$ be a model in which $\psi$ is satisfied. Then, for any primitive information source $db$, for any literals $l$ and for any $w \in W$ we have: $val(l) \in f_{db}(w) \implies val(-l) \not\in f_{db}(w)$.

**Lemma 1.3** Let $MS$ be a multi-set of valuations of literals such that for any literal $l$, $val(l) \in MS \implies val(-l) \not\in MS$. Then $min_{\leq MS}(W) = \forall l \in MS \exists l : val(l)$.

**Lemma 1.4** Let $l$ be a literal, $i$ an integer and $db$ an information source, primitive or not. We have: $\vdash MF \psi \rightarrow B_{db}^i l \iff \vdash MF \psi \rightarrow B_{db}^i l$ and $\vdash MF \psi \rightarrow \neg B_{db}^i l \iff \vdash MF \psi \rightarrow \neg B_{db}^i l$.

With these lemmas, we can now prove the ($\implies$) part. For doing so, we define for any $n \geq 1$:

**Definition** $\mathcal{H}(n)$ : for any formula $F$

$\vdash MF \psi \rightarrow B_{db_1, \ldots, db_n} F \iff \vdash MF \psi \rightarrow B_{db_1, \ldots, db_n}$

$\vdash MF \psi \rightarrow \neg B_{db_1, \ldots, db_n} F \iff \vdash MF \psi \rightarrow \neg B_{db_1, \ldots, db_n}$

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 $\vdash MF \psi \rightarrow B_{db} F$ then $\vdash MF \psi \rightarrow B_{db} F \Rightarrow \forall M \forall w \ M, w \vdash MF \psi \rightarrow B_{db} F$, so $\forall M \forall w \ M, w \vdash MF \psi \Rightarrow M, w \vdash MF B_{db} F$.

Let $M$ be a $MF$-model which satisfies $\psi$. 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, \(\text{val}(l) \in f_{db}(w) \Rightarrow \text{val}(-l) \notin f_{db}(w)\). Thus, by lemma 1.3, \(\min_{l \in f_{db}(w)} W = \text{val}(\bigwedge_{l \in f_{db}(w)} l)\). But \(g_{db}(w) = \min_{l \in f_{db}(w)} W\), so
\[
g_{db}(w) = \text{val}(\bigwedge_{l \in f_{db}(w)} l).
\]
Moreover, \(M, w \models MF B_{db} F \Rightarrow g_{db}(w) \subseteq \text{val}(F)\), so \(\text{val}(\bigwedge_{l \in f_{db}(w)} l) \subseteq \text{val}(F)\) and \(\models MF \bigwedge_{l \in f_{db}(w)} l \rightarrow F\).

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

\[
l' \in db \quad \Rightarrow \quad \neg l' \notin db \text{ because } db \text{ is consistent}
\]

\[
\models MF \psi \rightarrow B_{db}^0 l' \quad \text{and} \quad \models MF \psi \rightarrow B_{db}^0 \neg l'
\]

axiom schemata \(A4\)

\[
\models \psi \rightarrow B_{db}^0 l'
\]

(b) if \(F \equiv l_1 \lor \ldots \lor l_m\) then if \(F \equiv \top\), then \(\models MF F\), so \(\models MF B_{db} F\), thus \(\models MF \psi \rightarrow B_{db} F\).

If \(F \neq \top\), then \(\models MF \bigwedge_{l \in f_{db}(w)} l \rightarrow l_1 \lor \ldots \lor l_m\). Therefore, there is \(k \in \{1 \ldots m\}\) so that \(\models MF \bigwedge_{l \in f_{db}(w)} l \rightarrow l_k\). This case is the same as case 1.a.

(c) if \(F\) is a conjunction of disjunctions, that is, to say \(F \equiv \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \lor \ldots \lor l_{k_m})\), then \(\models MF \bigwedge_{l \in f_{db}(w)} l \rightarrow \bigwedge_{k \in \{1 \ldots n\}} (l_{k_1} \lor \ldots \lor l_{k_m})\) so \(\forall k \in \{1 \ldots n\}\)

\[
\models MF \bigwedge_{l \in f_{db}(w)} l \rightarrow l_{k_1} \lor \ldots \lor l_{k_m}\]. 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 \(g_{db}(w) = \text{val}(\bigwedge_{l \in f_{db}(w)} l)\) for any \(MF\)-model \(M\) of \(\psi\) and any world \(w\) of \(W\) by the same method.

\[
M, w \models MF \neg B_{db} F \models MF \iff g_{db}(w) \notin \text{val}(F)
\]

\[
\iff \models \left( \bigwedge_{l \in f_{db}(w)} l \right) \rightarrow F
\]

(a) if \(F\) is a literal \(l'\), then \(\models MF \left( \bigwedge_{l \in f_{db}(w)} l \right) \rightarrow l'\). In this case, \(l' \notin db\)

(because \(db\) is consistent).

\[
l' \notin db \quad \Rightarrow \quad \models MF \psi \rightarrow B_{db}^0 \neg l'
\]

There are two different cases:

- either \(\models MF \psi \rightarrow B_{db}^0 \neg l'\) and then by axiom schemata \(A6\), \(\models MF \psi \rightarrow \neg B_{db}^0 l'\),

\(\psi \rightarrow \neg B_{db}^0 l'\).
• or $\vdash MF \psi \rightarrow B_{db}^i \neg l'$ with $i > 0$ and:

axiom schemata A5 $\Rightarrow$ $\vdash MF \psi \rightarrow B_{db} \neg l'$

axiom schemata A1 $\Rightarrow$ $\vdash MF \psi \rightarrow \neg B_{db} l'$

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

$\not\vdash (\bigwedge_{l \in \mathcal{db}} l) \rightarrow F \quad \Rightarrow \quad \forall i \in \{1 \ldots m\} \ l_i \not\in \mathcal{db}$

$\vdash MF \psi \rightarrow B_{db}^0 l_1 \land \ldots \land B_{db}^0 l_m$

same proof as case 2.a $\Rightarrow$ $\vdash MF \psi \rightarrow \neg B_{db} l_1 \land \ldots \land \neg B_{db} l_m$

$\vdash MF \psi \rightarrow \neg (B_{db} l_1 \lor \ldots \lor B_{db} l_m)$

axiom schemata A7 $\Rightarrow$ $\vdash MF \psi \rightarrow \neg B_{db} (l_1 \lor \ldots \lor l_m)$

(c) if $F$ is a conjunction of clauses, i.e. $F \equiv \bigwedge_{k \in \{1 \ldots n\}} (l_k \lor \ldots \lor l_{km})$,

then:

$\not\vdash (\bigwedge_{l \in \mathcal{db}} l) \rightarrow F \quad \Rightarrow \quad \exists i \in \{1 \ldots n\} \ \not\vdash_{MF} (\bigwedge_{l \in \mathcal{db}} l) \rightarrow (l_{ki} \lor \ldots \lor l_{km})$

by the previous proof $\Rightarrow \exists i \in \{1 \ldots n\} \ \vdash_{MF} \psi \rightarrow \neg B_{db} (l_{ki} \lor \ldots \lor l_{km})$

$\Rightarrow \vdash_{MF} \psi \rightarrow \bigvee_{k \in \{1 \ldots n\}} \neg B_{db} (l_{ki} \lor \ldots \lor l_{km})$

$\Rightarrow \vdash_{MF} \psi \rightarrow \neg (\bigwedge_{k \in \{1 \ldots n\}} B_{db} (l_{ki} \lor \ldots \lor l_{km}))$

axiom sch. A1 et A2 $\Rightarrow \vdash_{MF} \psi \rightarrow \neg B_{db} (\bigwedge_{k \in \{1 \ldots n\}} (l_{ki} \lor \ldots \lor l_{km}))$

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

$\vdash MF \psi \rightarrow B_{db} F \quad \Rightarrow \quad \vdash MF \psi \rightarrow B_{db} F$

$\vdash MF \psi \rightarrow \neg B_{db} F \quad \Rightarrow \quad \vdash MF \psi \rightarrow \neg B_{db} F$

$\mathcal{H}(1)$ 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 $\vdash MF \psi \rightarrow B_{db \oplus db_{n+1}} F$.

(a) if $F$ is a literal $l$ of $L$ then for any $MF$-model $M$ of $\psi$ and for any world $w$, $g_{db \oplus db_{n+1}}(w) \subset val(l)$.
Let $i$ and $j$ be the two integers such that $\text{val}(l) \in^{i} f_{db*dn_{n+1}}(w)$ and $\text{val}(-l) \in^{j} f_{db*dn_{n+1}}(w)$. 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:

$$f_{db*dn_{n+1}}(w) = \left( \frac{\text{val}(l) \ldots \text{val}(l)}{i \text{ times}} \right) \left( \frac{\text{val}(-l) \ldots \text{val}(-l)}{j \text{ times}} \right) \left( \frac{\text{val}(l_1) \ldots \text{val}(l_m)}{\forall k \in \{1 \ldots m\} \ l_k \neq l} \right)$$

So:

$$dsum(w_1, f_{db*dn_{n+1}}(w)) = 0 \ast j + 1 \ast i + \sum_{k \in \{1 \ldots m\}} \min_{w' \in f_{d1*dn_{n+1}}(w)} d(w_1, w')$$

$$= i$$

$$dsum(w_2, f_{db*dn_{n+1}}(w)) = 1 \ast j + 0 \ast i + \sum_{k \in \{1 \ldots m\}} \min_{w' \in f_{d1*dn_{n+1}}(w)} d(w_2, w')$$

$$= j$$

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

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

So $\exists i \exists j \ i + j \geq 0$ such that $\text{val}(l) \in^{i} f_{db*dn_{n+1}}(w)$ and $\text{val}(-l) \in^{j} f_{db*dn_{n+1}}(w)$.

By building of $f_{db*dn_{n+1}}(w)$:

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

Therefore:

$$\begin{align*}
\text{lemma 1.4} & \Rightarrow \begin{cases}
\models MF \psi \rightarrow B^{i_1}_{db} l \\
\models MF \psi \rightarrow B^{i_2}_{db} \neg l
\end{cases} \text{ and } \begin{cases}
\models MF \psi \rightarrow B^{j_1}_{dn_{n+1}} l \\
\models MF \psi \rightarrow B^{j_2}_{dn_{n+1}} \neg l
\end{cases} \\
\text{axiom sch. A4} & \Rightarrow \begin{cases}
\vdash MF \psi \rightarrow B^{i_1}_{db} l \\
\vdash MF \psi \rightarrow B^{i_2}_{db} \neg l
\end{cases} \text{ and } \begin{cases}
\vdash MF \psi \rightarrow B^{j_1}_{dn_{n+1}} l \\
\vdash MF \psi \rightarrow B^{j_2}_{dn_{n+1}} \neg l
\end{cases} \\
\text{axiom sch. A5} & \Rightarrow \begin{cases}
\vdash MF \psi \rightarrow B_{db*dn_{n+1}} l \text{ (since } i > j) \\
\vdash MF \psi \rightarrow B_{db*dn_{n+1}} \neg l
\end{cases}
\end{align*}$$

(b) if $F$ is a disjunction of literals, i.e. $F \equiv l_1 \lor \ldots \lor l_m$, then if $F \equiv \top$, $\vdash MF F$, so $\vdash MF B_{db*dn_{n+1}} F$ (by (Nec)), and then $\vdash MF \psi \rightarrow B_{db*dn_{n+1}} F$. 23
If $F \not\equiv \top$, then for all $w \in W$ $f_{d^*d_{n+1}}(w)$ is such that $val(l) \in f_{d^*d_{n+1}}(w) \implies val'(l) \not\in f_{d^*d_{n+1}}(w)$. So, by lemma 1.3, $g_{d^*d_{n+1}}(w) = \min_{f_{d^*d_{n+1}}(w)} W$ is a conjunction of literals. But $g_{d^*d_{n+1}}(w) \subseteq val(F)$, so $g_{d^*d_{n+1}}(w) \equiv val(l_1 \lor \ldots \lor l_m)$. In this case, $\exists k \in \{1, \ldots, m\}$ such that $M \models_{MF} B_{d^*d_{n+1}} l_k$. This is the same case as before and then $\vdash_{MF} \psi \rightarrow B_{d^*d_{n+1}} l_k$. By axiom schemata A2, $
abla_{MF} \psi \rightarrow B_{d^*d_{n+1}} l_1 \lor \ldots \lor B_{d^*d_{n+1}} l_m$.

(c) if $F$ is a conjunction of clauses, i.e. $F \equiv \bigwedge_{k \in \{1, \ldots, n\}} (l_k \lor \ldots \lor l_m)$, then $\vdash_{MF} \psi \rightarrow B_{d^*d_{n+1}} l_k \lor \ldots \lor B_{d^*d_{n+1}} l_m$ and this is the same case as previously because of the $KD$ property of $B_{d^*d_{n+1}}$.

2. assume that $\vdash_{MF} \psi \rightarrow \neg B_{d^*d_{n+1}} F$.

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

For any $MF$-model $M$ of $\psi$ and for any world $w$, $g_{d^*d_{n+1}}(w) \not\subseteq val(l)$. Thus $\exists i \exists j \geq i \geq 0$ such that $val(l) \in f_{d^*d_{n+1}}(w)$ and $val(l^i) \in f_{d^*d_{n+1}}(w)$.

By building $f_{d^*d_{n+1}}(w)$, we can remark that:
- $\exists i_1 \exists i_2$ such that $i_1 + i_2 = i$, $val(l) \in f_{d^*d_{n+1}}$ and $val(l^i) \in f_{d^*d_{n+1}}$;
- $\exists j_1 \exists j_2$ such that $j_1 + j_2 = i$, $val(l) \in f_{d^*d_{n+1}}$ and $val(l^i) \in f_{d^*d_{n+1}}$.

So:

$$\begin{aligned}
\vdash_{MF} \psi &\rightarrow B_{d^*d_{n+1}}^{i_1} l \\
\vdash_{MF} \psi &\rightarrow B_{d^*d_{n+1}}^{j_1} \neg l
\end{aligned}$$

Lemma 1.4 $\Rightarrow$

$$\begin{aligned}
\vdash_{MF} \psi &\rightarrow B_{d^*d_{n+1}}^{i_1} l \\
\vdash_{MF} \psi &\rightarrow B_{d^*d_{n+1}}^{j_1} \neg l
\end{aligned}$$

A4 $\Rightarrow$

$$\begin{aligned}
\vdash_{MF} \psi &\rightarrow B_{d^*d_{n+1}}^{i_2} l \\
\vdash_{MF} \psi &\rightarrow B_{d^*d_{n+1}}^{j_2} \neg l
\end{aligned}$$

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

There are two cases:

- either $i = j$ and then by axiom A6, $\vdash_{MF} \psi \rightarrow \neg B_{d^*d_{n+1}}^{i_1} l$;
- either $j > i$ and then by axiom A5, $\vdash_{MF} \psi \rightarrow B_{d^*d_{n+1}}^{j_1} \neg l$. By axiom A1 $\vdash_{MF} \psi \rightarrow \neg B_{d^*d_{n+1}}^{i_1} l$.

The property is true when $F$ is a literal.

(b) if $F$ is a clause, i.e. $F \equiv l_1 \lor \ldots \lor l_m$, then for any $MF$-model $M$ of $\psi$, $M \models_{MF} \neg B_{d^*d_{n+1}} (l_1 \lor \ldots \lor l_m)$. 

24
As axiom schemata (A0), (A1) and (A2) are valid, \( \models_{MF} B_{dbs \cdot dbn+1}(l_1) \lor \ldots \lor B_{dbs \cdot dbn+1}(l_m) \rightarrow B_{dbs \cdot dbn+1}(l_1 \lor \ldots \lor l_m) \). So \( \models_{MF} \neg B_{dbs \cdot dbn+1}(l_1 \lor \ldots \lor l_m) \rightarrow B_{dbs \cdot dbn+1}(l_1) \lor \ldots \lor B_{dbs \cdot dbn+1}(l_m) \).

Thus \( \forall k \in \{1 \ldots m\} \models_{MF} \neg B_{dbs \cdot dbn+1}(l_k) \). By the previous proof, as \( M \) is a model of \( \psi \), \( \forall k \in \{1 \ldots m\} \models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1}(l_k) \). By axiom A7, \( \models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1}(l_1 \lor \ldots \lor l_m) \).

(c) if \( F \) is a conjunction of clauses, i.e. \( F \equiv \bigwedge_{k \in \{1 \ldots n\}} (l_k \lor \ldots \lor l_{km}) \),

\[
\models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1} \bigwedge_{k \in \{1 \ldots n\}} (l_k \lor \ldots \lor l_{km})
\]

Let \( M \) be a \( MF \)-model of \( \psi \), then \( M \models_{MF} \neg B_{dbs \cdot dbn+1} \bigwedge_{k \in \{1 \ldots n\}} (l_k \lor \ldots \lor l_{km}) \), thus \( \exists k \in \{1 \ldots m\} \models_{MF} \neg B_{dbs \cdot dbn+1}(l_k \lor \ldots \lor l_{km}) \).

So:

\[
\exists k \in \{1 \ldots m\} \models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1}(l_k \lor \ldots \lor l_{km})
\]

\[
\models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1}(l_k \lor \ldots \lor l_{km})
\]

\[
\models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1} \bigwedge_{k \in \{1 \ldots m\}} (l_k \lor \ldots \lor l_{km})
\]

\[
A1 \text{ and } A2 \quad \models_{MF} \psi \rightarrow \neg B_{dbs \cdot dbn+1} \bigwedge_{k \in \{1 \ldots m\}} (l_k \lor \ldots \lor l_{km})
\]

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

\( \Box \).

**Proposition 2.** Let \( \psi \) be the formula previously defined. Let \( F \) be a formula of \( L \) and \( db \) an information source. Then:

\[
\models_{MF} \psi \rightarrow B_{db} F \iff \models_{MF} \psi \rightarrow \neg B_{d} F
\]

**Proof.**

For proving \( \iff \), we first prove the following lemmas:

**Lemma 2.1.** Let \( M_1 = \langle W_1, val_1, R_1, B_1 \rangle \) and \( M_2 = \langle W_2, val_2, R_2, B_2 \rangle \) 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 \( M_1 = \langle W_1, val_1, R_1, B_1 \rangle \) and \( M_2 = \langle W_2, val_2, R_2, B_2 \rangle \) be two \( MF \)-models. \( \forall w_1 \in W_1 \exists w_2 \in W_2 \) such that \( \forall F \ w_1 \in val_1(F) \iff \models_{MF} \psi \rightarrow \neg B_{d} F \)

25
\( w_2 \in \text{val}_2(F) \).

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 \( \psi \). Then \( M_1 \models B_{dl}F \iff M_2 \models B_{dl}F \).

We prove lemma 2.3 by using lemma 2.1 and 2.2.

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

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

- either \( F \) is a literal \( l \) and:

\[
\forall M \ M \models \psi \implies \exists w \in W \ M, w \not\models B_{dl}l \\
\forall M \ M \models \psi \implies \exists w \in W \ \exists w' \in g_{dl}(w) \ u' \not\in \text{val}(l) \\
\forall M \ M \models \psi \implies \forall w \in W \ \exists w' \in g_{dl}(w) \ u' \not\in \text{val}(l) \text{ by lemma 2.4} \\
\forall M \ M \models \psi \implies \forall w \in W \ M, w \not\models B_{dl}l \\
\quad \implies \models \psi \to \neg B_{dl}l
\]

- either \( F \) is a clause \( l_1 \lor \ldots \lor l_n \) and:

\[
\forall M \ M \models \psi \implies \exists w \in W \ M, w \not\models B_{dl}(l_1 \lor \ldots \lor l_n) \\
\forall M \ M \models \psi \implies \exists w \in W \ \exists w' \in g_{dl}(w) \ u' \not\models l_1 \lor \ldots \lor l_n \\
\forall M \ M \models \psi \implies \forall w \in W \ \exists w' \in g_{dl}(w) \ \forall i \in \{1, \ldots, n\} \ u' \not\models l_i \\
\forall M \ M \models \psi \implies \forall w \in W \ \exists w' \in g_{dl}(w) \ \forall i \in \{1, \ldots, n\} \ u' \not\models \text{val}(l_i) \\
\forall M \ M \models \psi \implies \forall w \in W \ \forall i \in \{1, \ldots, n\} \ w \not\models B_{dl}(l_i) \text{ by lemma 2.4} \\
\forall M \ M \models \psi \implies \forall w \in W \ M, w \not\models B_{dl}(l_1 \lor \ldots \lor B_{dl}(l_n)) \\
\forall M \ M \models \psi \implies \forall w \in W \ M, w \models \neg (B_{dl}(l_1) \lor \ldots \lor \neg B_{dl}(l_n)) \\
\forall M \ M \models \psi \implies \forall w \in W \ M, w \models \neg B_{dl}(l_1 \lor \ldots \lor l_n) \text{ as axiom schemata (A7) is sound} \\
\quad \implies \models \psi \to \neg B_{dl}(l_1 \lor \ldots \lor l_n)
\]

26
• either \( F \) is a conjunction of clauses \( C_1 \land \ldots \land C_m \) and:

\[
\forall M \ M \models \psi \implies \exists w \in W \; M, w \not\models B_{db}(C_1 \land \ldots \land C_m)
\]

\[
\forall M \ M \models \psi \implies \exists w \in W \; \exists w' \in g_{db}(w) \; w' \not\models C_i \land \ldots \land C_m
\]

\[
\forall M \ M \models \psi \implies \exists w \in W \; \exists w' \in g_{db}(w) \; \exists i \in \{1, \ldots, m\} \; w' \not\models C_i
\]

\[
\forall M \ M \models \psi \implies \forall w \in W \; \exists \exists i \in \{1, \ldots, n\} \; w \not\models B_{db}(C_i)
\]

by lemma 2.4

\[
\forall M \ M \models \psi \implies \forall w \in W \; w \not\models B_{db}(C_1 \land \ldots \land B_{db}(C_m))
\]

\[
\forall M \ M \models \psi \implies M \models \neg B_{db}(C_1 \land \ldots \land C_m)
\]

as axiom schematas

(A0), (A1) and (A2) are sound

\[
\models \psi \rightarrow \neg B_{db}(C_1 \land \ldots \land C_m)
\]

For proving \( \models \), 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 \( \psi \), \( M \models_{MF} B_{db}F \) and \( M \models_{MF} \neg B_{db}F \). This is impossible (because \( \psi \) has at least one model), so \( \vdash_{MF} \psi \rightarrow \neg B_{db}F \implies \models_{MF} \psi \rightarrow B_{db}F \).

\( \Box \).

**Proposition 3.** Let \( db_1 \ldots db_n \) 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:

\[
\vdash \psi \rightarrow B_{\{\cdots db_i \cdots db_n\}} \iff \Delta_{\cdot\cdot}(\{db_1 \ldots db_n\}) \models F
\]

**Proof.**

We first notice that if \( \equiv \equiv_{MF} \psi \rightarrow B_{db} \) is a \( MF \)-model, then for any world \( w \) in \( W \), there is an interpretation \( u' \) of \( L \) such that \( \{l : u' \models l\} = \{l : w \in val(l)\} \) (see lemmas 2.1 and 2.2). And if \( u' \) is an interpretation of \( L \), then there is a world \( w \) in \( W \) such that \( \{l : u' \models l\} = \{l : w \in val(l)\} \). 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 \( \forall i = 1 \ldots n \; db_i = \{l_1^i \ldots l_m^i\} \).

Let \( \equiv \equiv_{MF} \psi \rightarrow B_{db} \) is a \( MF \)-model which satisfies \( \psi \). Let \( w \) be a world in \( W \) and \( w' \) the interpretation of \( L \) previously characterized. We prove that:

\[
distance_{\cdot\cdot}(w, [\ldots val(l_1^1) \ldots val(l_2^1) \ldots]) = distance_{\cdot\cdot}(w', [db_1 \ldots db_n])
\]

This shows that the \( W \) worlds which are minimal according to the order induced by the distance \( \distance_{\cdot\cdot} \) correspond to the interpretations of \( L \) which are minimal according to the order induced by the distance \( \distance_{\cdot\cdot} \).

\( \Box \).

**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 $B(db,l)$ if $\models_M \psi \rightarrow B_{db} l$

(2) PROLOG fails in proving $B(db,l)$ if $\models_M \psi \rightarrow \neg B_{db} l$

Proof.
For proving (1), we first prove that PROLOG succeeds in proving $R(db,l,i)$ if $\models_M \psi \rightarrow B^2_{db} l$. 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.

\[ \square \]

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

Proof
The proof of $\iff$ is obvious.

Proof of $\Rightarrow$
Let $l_1 \ldots l_n$ be some ground literals of $LO$ such that $l_1 \lor \ldots \lor l_n$ is not a tautology and $EDB \cup IDB \models l_1 \lor \ldots \lor l_n$.

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 $EDB \cup IDB \models l_{i_1} \land \ldots \land l_{i_m}$ (i.e. $\forall j \in \{1, \ldots, n\} - \{i_1, \ldots, i_m\}$) $EDB \cup IDB \models \neg l_j$. Since $EDB \cup IDB \models l_{i_1} \land \ldots \land l_{i_m}$, $\forall i \in \{i_1, \ldots, i_m\} \ \exists M_i$ Herbrand model of $EDB \cup IDB$ such that $H M_i \models l_i$.

Moreover, notice that $m > 1$, else $\forall H M$ Herbrand model of $EDB \cup IDB \ H M \models l_i$, which contradicts (hyp).

On the other hand, $\forall i \in \{i_1, \ldots, i_m\} \ \exists H M_i$ Herbrand model of $EDB \cup IDB$ such that $H M_i \models \neg l_i$, 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).

$l_{i_1} \land \ldots \land l_{i_m}$ is not a tautology, so $\forall i \in \{i_1, \ldots, i_m\} \ \forall j \in \{\{i_1, \ldots, i_m\} - \{i\}\} \ l_i \not\models \neg l_j$. Thus, $\neg l_{i_1} \land \ldots \land \neg l_{i_m}$ is satisfiable.

In this case, by definition 10, as $< EDB, IDB >$ is equivalent to a set of ground literals, $\exists H M_{i_0}$ Herbrand model of $EDB \cup IDB$ such that $H M_{i_0} \models \neg l_{i_1} \land \ldots \land \neg l_{i_m}$. Thus, by 1.1, $H M_{i_0} \models \neg l_1 \land \ldots \land \neg l_n$. This contradicts the fact that $EDB \cup IDB \models l_1 \lor \ldots \lor l_n$, so (hyp) is false.

\[ \square \]

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

Proof.
Proof of $\Rightarrow$)
We prove in the same time the two following propositions:

(1) \( \text{PROLOG proves } B_{\text{exp}}(db, l) \implies EDB \cup IDB \models l \) and

(2) \( \text{PROLOG proves } B_{\text{conj}}(db, l_1 \land \ldots \land l_n) \implies EDB \cup IDB \models l_1 \land \ldots \land l_m \)

Let us denote:

\( \mathcal{H}(n): \) PROLOG proves \( B_{\text{exp}}(db, l) \) in \( n \) steps \( \implies EDB \cup IDB \models l \)

PROLOG proves \( B_{\text{conj}}(db, l_1 \land \ldots \land l_m) \) in \( n + 1 \) steps \( \implies EDB \cup IDB \models l_1 \land \ldots \land l_m \)

We will prove part \( \implies \) 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 \( B_{\text{exp}}(db, l) \) or \( B_{\text{conj}}(db, l_1 \land \ldots \land l_m) \) in a finite number of steps.

Let us suppose that PROLOG cannot prove \( B_{\text{exp}}(db, l) \) or \( B_{\text{conj}}(db, l_1 \land \ldots \land l_m) \) in a finite number of steps. In this case, it means that PROLOG tries to prove some formula \( B_{\text{exp}}(db, l') \) or \( B_{\text{conj}}(db, l'_1 \land \ldots \land l'_{m}) \) 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.

\textbf{Proof of } \( \mathcal{H}(2) \): PROLOG proves \( B_{\text{exp}}(db, l) \) 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 \( B_{\text{conj}}(db, l_1 \land \ldots \land l_m) \) in three steps. The only axiom PROLOG can use for first step is (1.6). So PROLOG has to prove:

- \( B_{\text{conj}}(db, l_2 \land \ldots \land l_m) \) in two steps. The only axiom PROLOG can use is (1.5), so in fact \( l_2 \land \ldots \land l_m \equiv \text{nil} \).

- \( B_{\text{exp}}(db, l) \) in two steps. By the previous proof, \( EDB \cup IDB \models l_1 \).

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

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

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

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

- \( B_{\text{conj}}(db, l_2 \land \ldots \land l_m) \) in three steps. As \( \mathcal{H}(2) \) is true, \( EDB \cup IDB \models l_2 \land \ldots \land l_m \).
• $B_{exp}(db,l)$ 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, $EDB \cup IDB \models l_1 \land \ldots \land l_m$ 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 $B_{conj}(db,l_1 \land \ldots \land l_m)$ 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 $B_{conj}(db,l_2 \land \ldots \land l_m)$ in $n+1$ steps. As $\mathcal{H}(n)$ is true, $EDB \cup IDB \models l_2 \land \ldots \land l_m$.

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

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

Proof of $\iff$

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 $C$ of $IDB$ containing the variables $x_{i_1}, \ldots , x_{i_k}$, we will call a ground instance of $C$ any formula $\mathcal{F}$ of $LO$ such that $\mathcal{F} \equiv C[x_{i_1}/a_{j_1}, \ldots , x_{i_k}/a_{j_k}]$ (i.e. a clause in which each variable symbol is replaced by a constant symbol).

Let us define $\text{succ}$ the function such that for any set $E$ of ground literals of $LO$:

$$\text{succ}(E) = E \cup \{P_i(a_{i_1}, \ldots , a_{i_{m_i}}) \cup \neg P_{j_1}(a_{j_{i_1}, \ldots , a_{j_{i_{m_i}}}}) \land \ldots \land \neg P_{j_k}(a_{j_{i_1}, \ldots , a_{j_{i_{m_k}}}}) \rightarrow P_i(a_{i_1}, \ldots , a_{i_{m_i}}) \text{ is a ground instance of a clause of h(c) where c \in IDB and } \forall k \in \{j_1, \ldots , j_k\} \neg P_{j_k}(a_{j_{i_1}, \ldots , a_{j_{i_{m_k}}}}) \in E \}$$

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

$$\mathcal{T}_1(EDB) = EDB$$

$$\forall n \geq 1 \mathcal{T}_{n+1}(EDB) = \text{succ}(\mathcal{T}_n(EDB))$$

30
As $LO$ is a finite language, it is easy to prove that $\exists i_0 \geq 1$ such that $\lim_{n \to \infty} T_n(EBD) = T_{i_0}(EBD)$ ($\text{succ}$ is a growing series in a finite set). It is also obvious that for any ground literal $l$, $EBD \cup IDB \models l \iff l \in T_{i_0}(EBD)$.

Let us denote $T_{i_0}(EBD)$ as $T^1(EBD)$.

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

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

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

**Induction step for $K$:** let us suppose that for some $i \geq 1$ $K(i)$ is true. Let us suppose that $l \in T^{i+1}(EBD)$. In this case, there is a formula $P_{j_1}(a_{j_1,1}, \ldots, a_{j_1,i_j}) \land \ldots \land \neg P_{j_k}(a_{j_k,1}, \ldots, a_{j_k,i_k}) \rightarrow l$ which is an instance of a clause in $IDB$ such that $\forall k \in \{j_1, \ldots, j_k\}, \neg P_{j_k}(a_{j_k,1}, \ldots, a_{j_k,i_k}) \in T_i(EBD)$.

By axiom (1.2), PROLOG proves $IDB(db, \neg P_{j_1}(a_{j_1,1}, \ldots, a_{j_1,i_j}) \land \ldots \land \neg P_{j_k}(a_{j_k,1}, \ldots, a_{j_k,i_k}) \rightarrow l)$.

As $\forall k \in \{j_1, \ldots, j_k\}, \neg P_{j_k}(a_{j_k,1}, \ldots, a_{j_k,i_k}) \in T_i(EBD)$ and $K(i)$ is true, PROLOG proves $B_{exp}(db, \neg P_{j_1}(a_{j_1,1}, \ldots, a_{j_1,i_j}))$ for any $k \in \{j_1, \ldots, j_k\}$. So, by axiom (1.5) and (1.6), PROLOG proves $B_{conj}(db, \neg P_{j_1}(a_{j_1,1}, \ldots, a_{j_1,i_j}) \land \ldots \land \neg P_{j_k}(a_{j_k,1}, \ldots, a_{j_k,i_k}))$.

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

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

\[\square\]