On the Compilation of Stratified Belief Bases under Linear and Possibilistic Logic Policies

Salem Benferhat
CRIL-CNRS, Université d'Artois
rue Jean Souvraz
62307 Lens Cedex. France.
benferhat@cril.univ-artois.fr

Safa Yahi
Université USTHB
BP 32 El-Alia Bab Ezzouar
16111 Algiers. Algeria.
yahi.safa@gmail.com

Habiba Drias
Institut National d'Informatique
BP 68M Oued Smar
16309 Algiers. Algeria.
h_drias@ini.dz

Abstract

Developing efficient approaches for reasoning under inconsistency is an important issue in many applications. Several methods have been proposed to compile, possibly inconsistent, weighted or stratified bases. This paper focuses on the well-known linear order and possibilistic logic strategies. It provides a way for compiling a stratified belief base in order to be able to process inference from it in polynomial time. The resulting extra compilation cost is very low. In particular, the number of additional variables, that are added to original stratified bases, corresponds exactly to the number of priority levels existing in the base. Moreover, our compilation approach allows an efficient computation of weighted possibilistic conclusions and possibilistic conditioning.

1 Introduction

Integrating pieces of information coming from different, possibly conflicting sources may lead to reason with inconsistent formulae. In this case, classical inference can not be used to exploit the given belief base, since from an inconsistent base every formula can be inferred (*ex falso quodlibet sequitur* principle).

Many approaches have been proposed in order to reason under inconsistency without trivialization. While some of them consist in weakening the inference relation such as *paraconsistant logics*, others weaken the available beliefs like the well-known *coherence based-approaches* [Rescher and Manor, 1970].

Coherence-based approaches can be characterized by the two following steps: i) give up some formulas of the inconsistent belief base Σ in order to restore its consistency; this operation results into one or several consistent subbases of Σ; and ii) use classical entailment on these consistent subbases to define plausible conclusions from the knowledge base Σ. When priorities attached to pieces of knowledge are available, the task of coping with inconsistency is simplified, because priorities give information about how to solve conflicts.

Examples of prioritized coherence approaches which select one consistent subbase are the possibilistic logic approach [Dubois *et al.*, 1994], Williams' adjustment approach [Williams, 1995], and linear order approach [Nebel, 1994]. Examples of approaches which use several consistent subbases are Rescher's notion of acceptable subbases [Rescher, 1976], Brewka's preferred sub-theories [Brewka, 1989] and the lexicographical system [Benferhat *et al.*, 1993; Lehmann, 1995].

In this paper, we mainly focus on the linear order and on possibilistic logic policies. Given a stratified belief base $\Sigma = (S_1, \ldots, S_k)$ (S_1 contains the most prioritized formulas, while the stratum S_k contains the least prioritized formulas), possibilistic logic only takes into account the first i consistent strata, and stops when inconsistency is met. Nebel [Nebel, 1994] has elaborated a less liberal way to select one consistent subbase. When inconsistency occurs at some strata k, we give up the whole stratum (concerned by inconsistency), but we continue to add strata with lower certainty if consistency is preserved.

From a practical point of view, reasoning under inconsistency is a hard problem [Nebel, 1998]. Indeed, deciding the inference from a stratified belief base with respect to the possibilistic policy is a $\Delta_2^p[O(log\ n)]$-complete problem while deciding the inference from it with respect to the linear order policy is Δ_2^p-complete.

To address this problem, one can use knowledge compilation. Knowledge compilation is a key direction of research in Artificial Intelligence [Selman and H.Kautz, 1996; Cadoli and Donini, 1997; Darwiche and Marquis, 2002]. It consists in preprocessing off-line the knowledge base in order to make the inference from it easier on-line.

A key issue, in possibilistic logic, is how to determine the level of uncertainty associated with formulas that can be derived from an inconsistent knowledge base. Another important issue is how to compute "possibilistic conditioning". Namely, like in possibilistic networks, given some evidence, often provided in the form of a conjunction of literals (a term), we need to evaluate the impact of this evidence on the certainty degrees of variables or more generally of formulas of knowledge bases.

In this paper, we propose an efficient way to compile stratified belief bases with respect to the linear order and possibilistic logic policies. In order to support possibilistic conditioning and the computation of weights associated with concluded formulas, we need to keep track and encode the weights associated with formulas. This is obtained by adding new variables, as it has been previously done in some existing works [Darwiche and Marquis, 2004; Coste-Marquis and Marquis, 2004; Benferhat and Prade, 2005].

The main advantage of our approach is the resulting extra compilation cost which is very low. Indeed, the number of additional variables needed for compiling stratified knowledge bases under linear order and possibilistic policy, is only equal to the number of strata in the belief base. In [Benferhat and Prade, 2006] a compilation method has been proposed to compile possibilistic bases (and not linear order policy). However, this approach needs to add binary clauses and is based on the idea of forgetting variables which limits the target compilation languages that can be exploited. Moreover, it does not support possibilistic conditioning.

The remainder of the paper is organized as follows. Section 2 introduces stratified belief bases and the corresponding inference relations w.r.t the linear order and possibilistic logic policies. We also define how to associate weights to the inferred conclusions and the notion of possibilistic conditioning. After a brief background on knowledge compilation in Section 3, we describe our compilation method in the case of linear order policy in Section 4. In Section 5, we adapt this method to compile possibilistic knowledge bases. We also, show how to handle weighted possibilistic inference relation and possibilistic conditioning. Section 6 provides some comparative studies and concludes the paper.

2 Stratified Belief Bases

We consider a finite propositional language denoted by L. $\models$ denotes classical consequence relation. Greek letters Φ, Ψ, ... represent formulas. $\top$ and $\bot$ denote the constants true and false respectively.

A stratified belief base is a set of formulae equipped with a total pre-order. Formally:

Definition 1 *A stratified belief base (SBB for short)Σ is a finite sequence (S_1, ..., S_k) where S_i ($i = 1, \ldots, k$) is a stratum containing propositional formulas having the same priority level i and which are more reliable than formulas of the stratum S_j for $j > i$.*

According to coherence-based approaches, the inference from a stratified belief base is considered as a two step process consisting first in generating some preferred consistent subbases of Σ and then using classical inference from some of them depending on the entailment principle used.

Many policies for the selection of subbases can be considered, especially the *possibilistic policy* [Dubois *et al.*, 1994], *the linear order policy* [Nebel, 1994], *the inclusion preference policy* [Brewka, 1989] and *the lexicographical policy* [Benferhat *et al.*, 1993]. Among them, we focus on the possibilistic and the linear order policies.

The idea of the possibilistic logic policy is to start with formulas having the highest priority level and to propagate as many prioritized formulas as possible while maintaining consistency. We stop at the first priority level where we meet inconsistency and the formulas at this level and below are ignored. Formally:

Definition 2 *Let $\Sigma = (S_1, \ldots, S_k)$ be a SBB. The set Σ_{PO} of all the preferred subbases of Σ w.r.t the possibilistic policy is the singleton $\{\bigcup_{i=1}^{s-1} S_i\}$, where s is the smallest index s.t. $\bigcup_{i=1}^{s} S_i$ is inconsistent. If Σ is consistent, then $\Sigma_{PO} = \Sigma$.*

As to the linear order policy which keeps more information, it does not stop at the first level where it meets inconsistency: when an inconsistency is encountered, the current stratum is dropped, and the process moves to the next stratum and so on. Formally:

Definition 3 *Let $\Sigma = (S_1, \ldots, S_k)$ be a SBB. The set Σ_{LO} of all the preferred subbases of Σ w.r.t the linear order policy is the singleton $\{\bigcup_{i=1}^{k} S'_i\}$, where S'_i is defined by $S'_i = S_i$ if $S_i \cup \bigcup_{j=1}^{i-1} S'_j$ is consistent, $\emptyset$ otherwise.*

Example 1 *Let $\Sigma = (S_1, S_2, S_3, S_4)$ be such that:*
$S_1 = \{a \vee b\}$
$S_2 = \{\neg a, c\}$
$S_3 = \{\neg b, d \vee e\}$
$S_4 = \{b \vee d, e\}$.

- *$S_1 \cup S_2$ is consistent whereas $S_1 \cup S_2 \cup S_3$ is not. According to the possibilistic policy, we stop at the first stratum where we meet inconsistency thus $\Sigma_{PO} = S_1 \cup S_2 = \{a \vee b, \neg a, c\}$.*
- *As for the linear order policy, we delete S_3 and continue the treatment with the other strata. $S_1 \cup S_2 \cup S_4$ is consistent so $\Sigma_{LO} = S_1 \cup S_2 \cup S_4 = \{a \vee b, \neg a, c, b \vee d, e\}$.*

Definition 4 *Let $\Sigma = (S_1, \ldots, S_k)$ be a SBB and Ψ a formula.*

- *Ψ is a consequence of Σ w.r.t the possibilistic policy, denoted by $\Sigma \models_{PO} \Psi$, iff $\Sigma_{PO} \models \Psi$.*
- *Ψ is a consequence of Σ w.r.t the linear order policy, denoted by $\Sigma \models_{LO} \Psi$, iff $\Sigma_{LO} \models \Psi$.*

Example 2 *Let us consider again the stratified belief base Σ given in Example 1. For the first inference relation, we have for instance, $\Sigma \models_{PO} b$. As to the second one, we have for example $\Sigma \models_{LO} e$.*

It is easy to see that $\models_{LO}$ goes beyond $\models_{PO}$, namely for each propositional formula Ψ, if $\Sigma \models_{PO} \Psi$ then $\Sigma \models_{LO} \Psi$.

A key issue in possibilistic logic is how to compute the certainty degree associated with inferred formulas:

Definition 5 *Let $\Sigma = (S_1, \ldots, S_k)$ be a stratified belief base and Ψ be a formula. Ψ is a weighted consequence of Σ to a degree i, denoted by $\Sigma \models_{we} (\Psi, i)$, iff:*
a) $S_1 \cup \ldots \cup S_i$ is consistent;
b) $S_1 \cup \ldots \cup S_i \models \Psi$;
c) $\nexists j < i : S_1 \cup \ldots \cup S_j \models \Psi$.

Lastly, in possibilistic logic we are also interested in computing possibilistic conditioning. Namely,

Definition 6 *Let e be a consistent term. Then Ψ is a conclusion of conditioning Σ by e to a rank i iff $\Sigma \cup S_0 = \{e\} \models_{we} (\Psi, i)$, where S_0 is a new and high important stratum.*

Possibilistic conditioning inference is an important concept in possibilistic theory. It is, to some extent, the logical counterpart of possibilistic propagation algorithms in possibilistic networks.

Following [Nebel, 1998; Lang, 2000], deciding the possibilistic inference is $\Delta_2^p[O(log\ n)]$-complete : it needs at most $log_2\ n$ calls to SAT solver. As for the linear order one, it is more expensive from a computational point of view. Indeed, deciding it is a Δ_2^p-complete which means that it can be decided in polynomial time using NP oracles [Nebel, 1994]. We refer the reader to [Papadimitriou, 1994] for more details about computational complexity.

3 Knowledge Compilation

The key motivation of *Knowledge compilation* is that a knowledge base is not modified very often, and the same base is used to answer many queries (see [Cadoli and Donini, 1997] for a survey). So, the idea of knowledge compilation is to split query answering into two phases:

- first, the knowledge base is preprocessed to obtain an appropriate data structure. Such a phase is called *off-line* reasoning.
- the second phase which is called *on-line reasoning*, consists in answering queries using the data structure generated during the first phase.

A target compilation language is a class of formulas which is tractable for clause deduction at least. Stated otherwise, it is a language which permits at least achieving clausal deduction in polynomial time.

Initially, the well-known prime implicates language (PI for short) was the target compilation language on which most of the compilation methods focus. Recently, Darwiche and Marquis have considered in [Darwiche and Marquis, 2002] other target compilation languages. These languages are special cases of the *NNF* (Normal Negation Form) language obtained by imposing some properties. A NNF formula is a formula constructed with literals using only the conjunction and disjunction operators. As to the properties, one can list *decomposability*, *determinism*, *smoothness*, *decision*, *order*, etc.

The resulting target compilation languages are DNF, DNNF, d-DNNF, sd-DNNF, FBDD, OBDD, $OBDD_<$, MODS, PI and IP. Additionally, they are compared in terms of their spacial efficiency via the *succinctness* criteria and also in terms of the set of logical queries and transformations they support in polynomial time.

With the exception of PI, DNNF is the most succinct among all target compilation languages. In fact, it is known that PI is not more succinct than DNNF, but it is unknown whether DNNF is more succinct than PI. A sentence in DNNF (for Decomposable NNF) is a NNF sentence satisfying the decomposability property: for each conjunction C in the sentence, the conjuncts of C do not share variables [Darwiche, 2001].

4 Compiling Stratified Belief Bases under the Linear Order Policy

A classical way to compile a SBB under the linear order (resp. possibilistic logic) policy consists first in generating the preferred consistent subbase with respect to this policy. This step comes down to solve a Δ_2^P-complete (resp. $\Delta_2^p[O(log\ n)]$-complete) problem. Then we compile the resulting (classical) base into a target compilation language. We stress here that such an effort can not be exploited for handling the possibilistic logic policy. Namely, such approach is not appropriate to compute weighted possibilistic conclusions and possibilistic conditioning.

Our method relies on a propositional encoding of the SBB by adding new variables which the number is equal to the number of its strata. Then we proceed to compiling this propositional encoding into a target compilation language and then generating in polynomial time the base that corresponds to the linear order policy or even to the possibilistic one. The additional variables will be deleted from the final result.

4.1 Propositional Encoding of a Stratified Belief Base

Let $\Sigma = (S_1, \ldots, S_n)$ be a SBB. The first step in our compilation method consists in encoding the given base into a classical propositional base denoted by K_Σ. This encoding takes advantage of the one recently proposed in [Benferhat and Prade, 2005] for handling an extension of possibilistic logic.

In fact, for each stratum S_i, we will introduce a new propositional variable A_i. Intuitively, A_i means something like "the situation is A_i- abnormal". Hence, for each formula Φ_{ij} from the stratum S_i will correspond the propositional formula $\Phi_{ij} \vee A_i$ which can be read as "Φ_{ij} is true or the situation is A_i-abnormal".

In this paper, $(\Phi_{ij} \vee A_i)$ will be used as an encoding of the fact that the certainty degree or the priority rank of Φ_{ij} is A_i. The use of abnormal predicates have been used in several non-monotonic formalisms, e.g., [McCarthy, 1980].

Definition 7 *Let $\Sigma = (S_1, \ldots, S_k)$ be a SBB. The propositional encoding associated with Σ, denoted by K_Σ, is the propositional knowledge base defined by:*

$$K_\Sigma = \bigcup_{i=1}^{k} \{\bigcup_{\Phi_{ij} \in S_i} \{\Phi_{ij} \vee A_i\}\}$$

We note that generally, the number of strata in a SBB is very low. Hence, the cost of the propositional encoding we propose here is very low too.

Example 3 *Let us consider again the SBB Σ given in Example 1. Let A_1, A_2, A_3 and A_4 be four propositional variables associated with strata S_1, S_2, S_3 and S_4 respectively. Using the above definition, the propositional knowledge base associated with Σ is:*

$$K_\Sigma = \{a \vee b \vee A_1, \neg a \vee A_2, c \vee A_2, \neg b \vee A_3, d \vee e \vee A_3, b \vee d \vee A_4, e \vee A_4\}.$$

4.2 Conditioning

We review in this section, the operation of *conditioning* which will be useful for our approach.

Definition 8 *The conditioning of a propositional formula Ψ on a consistent term (a conjunction of literals)γ, denoted by CD(Ψ, γ), is the formula obtained from Ψ by substituting every literal of γ in Ψ by $\top$ if it is consistent with γ, by $\bot$ otherwise.*

Example 4 *For instance we have:*

- *The conditioning of the formula $\Psi = (\neg a \wedge \neg b) \vee (b \wedge c)$ on the term $\gamma = \neg a \wedge c$, $CD(\Psi, \gamma)$, gives $(\top \wedge \neg b) \vee (b \wedge \top)$.*
- *Conditioning the same formula on the term $a \wedge \neg b$, $CD(\Psi, a \wedge \neg b)$ gives $(\bot \wedge \top) \vee (\bot \wedge c)$.*

Following [Darwiche and Marquis, 2002], for any target compilation language C considered in [Darwiche and Marquis, 2002], C satisfies conditioning. This means that given a formula Ψ from C and any consistent term γ, we can construct in polytime a formula which belongs to C and is equivalent to $CD(\Psi, \gamma)$.

Remark 1 *In the following, in CD(Ψ, γ), γ is always a literal (an atom or a negated atom).*

4.3 Compiling Linear Based Knowledge Bases

Given a stratified belief base Σ, we define its compilation with respect to a given target compilation language C as shown by the following definition:

Definition 9 *Let $\Sigma = (S_1, \ldots, S_k)$ be a stratified belief base and K_Σ be its propositional encoding using Definition 7. Let C be any target compilation language. The compilation of Σ w.r.t. C, denoted by $Comp^C_\Sigma$, is the compilation of K_Σ into C, i.e., $C(K_\Sigma)$.*

We propose now the following algorithm which provides an efficient way to generate Σ_{LO} using $Comp^C_\Sigma$ and conditioning:

```
Algorithm 1: Computing $\Sigma_{LO}$
Data: $Comp^C_\Sigma$
Result: The compilation of $\Sigma_{LO}$
Begin
$K \leftarrow Comp^C_\Sigma$;
for $i \leftarrow 1$ to $k$ do
    if $K \models A_i$ then $K \leftarrow CD(K, A_i)$;
      else $K \leftarrow CD(K, \neg A_i)$;
    end
return $K$;
end
```

Proposition 1 *The propositional knowledge base returned by Algorithm 1 is equivalent to Σ_{LO} given by Definition 3.*

Proposition 1 shows the equivalence of the knowledge base returned by the Algorithm 1 and Σ_{LO}. Let us briefly explain the algorithm. It proceeds stratum per stratum and generates progressively the compiled base w.r.t. the linear order policy. The test in step 3 checks whether the stratum S_i is inconsistent (with previously accepted beliefs). If it is the case, then all formulas of such stratum are ignored by replacing $A_i \vee \Phi_{ij}$ by $\top \vee \Phi_{ij}$, which is equivalent to a tautology. If it is not the case, the Φ_{ij} are retained since $A_i \vee \Phi_{ij}$ is replaced by $\bot \vee \Phi_{ij}$ which is equivalent to Φ_{ij}.

Let us illustrate the above algorithm with the following example:

Example 5 *Let $\Sigma = (S_1, S_2, S_3)$ be such that*
$S_1 = \{\neg a \vee b, a\}$
$S_2 = \{\neg b, c\}$ *and* $S_3 = \{\neg b \vee \neg d\}$.
It is easy to check that:

$$\Sigma_{LO} = \{\neg a \vee b, a, \neg b \vee \neg d\}.$$

Let A_1, A_2 and A_3 be three propositional variables associated with S_1, S_2 and S_3 respectively.
$K_\Sigma = \{\neg a \vee b \vee A_1, a \vee A_1, \neg b \vee A_2, c \vee A_2, \neg b \vee \neg d \vee A_3\}$.
Let C = DNNF so let us compute $Comp^{DNNF}_\Sigma$, we get :
$K = Comp^{DNNF}_\Sigma = DNNF(K_\Sigma) = (\neg b \wedge A_1 \wedge (c \vee A_2)) \vee (b \wedge A_2 \wedge (a \vee A_1) \wedge (\neg d \vee A_3))$.

- *At the first iteration, we have $K \not\models A_1$ so*
  $K \leftarrow CD(K, \neg A_1) = b \wedge A_2 \wedge a \wedge (\neg d \vee A_3)$
- *At the second iteration, $K \models A_2$ implies that*
  $K \leftarrow CD(K, A_2) = b \wedge a \wedge (\neg d \vee A_3)$
- *At the third iteration, we have $K \not\models A_3$ thus*
  $K \leftarrow CD(K, \neg A_3) = b \wedge a \wedge \neg d$.

One can easily check that K is equivalent to Σ_{LO}.

Proposition 2 *Algorithm 1 is achieved in polynomial time in the size of $Comp^C_\Sigma$. Moreover, the propositional base returned by Algorithm 1 belongs to the target compilation language C.*

The corresponding proof is as follows. For all target compilation language C, C satisfies conditioning [Darwiche and Marquis, 2002]. This means that $CD(K, A_i)$ (or $CD(K, \neg A_i)$) can be achieved in polynomial time and K still belongs to C. In addition, since A_i is a clause, the deduction test $K \models A_i$ can be achieved in polynomial time. □

Since Σ_{LO} belongs to a target compilation language, the inference from a stratified belief base Σ interpreted under the linear order policy falls from Δ^p_2 down to P when queries are in a CNF form.

It is important to note that the cost induced here is only the one of the computation of $Comp^C_\Sigma$, i.e., the cost of the compilation of the propositional base K_Σ into the language C. The choice of the target compilation language is determined only via its concision. Hence, DNNF may be more interesting than the other languages in this case.

5 Computing Possibilistic Conclusions and Possibilistic Conditioning

5.1 Computing Possibilistic Inference

Compiling possibilistic knowledge bases has been recently achieved in [Benferhat and Prade, 2006]. In this section, we propose an alternative approach which offers several advantages with respect to the existing ones (see related works section). Contrary to the approach proposed in [Benferhat and Prade, 2006], our approach does not add binary clauses. Moreover, our approach is flexible since it supports all target compilation languages. Lastly, our approach allows the computation of possibilistic conditioning.

In fact, we suggest deriving another algorithm from Algorithm 1 where once we meet an inconsistency for the first time, we suppress the current stratum responsible of the inconsistency but also all the remaining strata.

```
Algorithm 2 : Computing $\Sigma_{PO}$
Data: $Comp_{\Sigma}^{C}$
Result: The compilation of $\Sigma_{PO}$
Begin
$K \leftarrow Comp_{\Sigma}^{C}$;
$i \leftarrow 1$ ;
while $K \nvDash A_i$ and $i \leq k$ do
    $K \leftarrow CD(K, \neg A_i)$;
    $i \leftarrow i + 1$;
    end
while $i \leq k$ do
    $K \leftarrow CD(K, A_i)$;
    $i \leftarrow i + 1$;
    end
 return $K$;
end
```

Proposition 3 *The propositional knowledge returned by the Algorithm 2 is equivalent to Σ_{PO} given by Definition 2 and belongs to C. Also, Algorithm 2 runs in polynomial time.*

Example 6 *Let us consider again the SBB Σ given in Example 5. We can easily check that: $\Sigma_{PO} = \{\neg a \vee b, a\}$. $Comp_{\Sigma}^{DNNF} = (\neg b \wedge A_1 \wedge (c \vee A_2)) \vee (b \wedge A_2 \wedge (a \vee A_1) \wedge (\neg d \vee A_3))$. Until the second iteration, namely until meeting an inconsistency, the process is the same as the linear order policy so $K = b \wedge a \wedge (\neg d \vee A_3)$. From here, all the remaining strata are ignored thus $K \leftarrow CD(K, A_3) = b \wedge a$. Again one can easily check that K is equivalent to Σ_{PO}.*

5.2 Compiling Weighted Consequences and Possibilistic Conditioning

With a simple adaptation, one can still exploit $Comp_{\Sigma}^{C}$ in order to decide the weighted possibilistic inference relation in polynomial time as shown by the following algorithm:

```
Algorithm 3 : Computing weighted conclusions
Data: $Comp_{\Sigma}^{C}$, a CNF formula $\Psi$
Result: a degree associated with $\Psi$
Begin
$K \leftarrow Comp_{\Sigma}^{C}$;
is_consequence $\leftarrow$ false;
$i \leftarrow 1$ ;
while $K \nvDash A_i$ and $i \leq k$ and is_consequence = false
    do
    $K \leftarrow CD(K, \neg A_i)$;
    if ($K \models \Psi$) then is_consequence $\leftarrow$ true;
      else $i \leftarrow i + 1$;
    end
if is_consequence = true
  then $\Psi$ is a consequence of $\Sigma$ with a degree $i$
  else $\Psi$ is not a consequence of $\Sigma$
end
```

One can check that $\Sigma \models_{we} (\Psi, i)$ using Definition 5 if and only if Ψ is a consequence of Σ to a degree i using Algorithm 3.

Example 7 *Let $\Sigma = (S_1, S_2, S_3)$ be such that:*
$S_1 = \{\neg a \vee b\}$, $S_2 = \{a, c\}$ and $S_3 = \{\neg b\}$.
Let us check if b is a possibilistic consequence of Σ and to what degree. We can check that $\Sigma \models_{we} (b, 2)$. Let A_1, A_2 and A_3 be three propositional variables associated respectively with the three strata in the knowledge base. K_Σ = {($\neg a \vee b \vee A_1$), ($a \vee A_2$), ($c \vee A_2$), ($\neg b \vee A_3$)}. Let i = 1 and C = DNNF.

$K = Comp_{\Sigma}^{DNNF} = (a \wedge (c \vee (\neg c \wedge A_2)) \wedge ((b \wedge A_3) \vee (b \wedge A_1))) \vee (\neg a \wedge A_2 \wedge (\neg b \vee (b \wedge A_3)))$.

- *At the first iteration, $K \nvDash A_1$. So, $K \leftarrow CD(K, \neg A_1) \equiv (a \wedge b \wedge A_3 \wedge (c \vee (\neg c \wedge A_2))) \vee (\neg a \wedge A_2 \wedge (\neg b \vee (b \wedge A_3)))$. $K \nvDash b$ so $i \leftarrow i + 1$.*
- *At the second iteration, $K \nvDash A_2$. So, $K \leftarrow CD(K, \neg A_2) \equiv a \wedge b \wedge A_3 \wedge c$. $K \models b$ so is_consequence $\leftarrow$ true which stops the loop. We deduce then that b is a possibilistic consequence of Σ to degree 2.*

Given a stratified belief base Σ that have been compiled in $Comp_{\Sigma}^{C}$. We show how we can exploit such a compilation effort in order to support the corresponding possibilistic conditioning by a given observation e in polynomial time.

It is enough to substitute in Algorithm 3 the line
"**while** $K \nvDash A_i$ **and** $i \leq k$ **and** is_consequence = false"
by :
"**while** $K \nvDash \neg e \vee A_i$ **and** $i \leq k$ **and** is_consequence = false".

Clearly, this adapted algorithm runs in polynomial time since K belongs to a target compilation language and since $\neg a \vee A_i$ is a clause. We can also check that it captures possibilistic conditioning.

6 Related Works and Conclusions

In this paper, we have proposed a method for compiling stratified belief bases with respect to the possibilistic and linear order policies. The cost of the proposed method is very low since the number of new variables that we introduce is equal to the number of the strata which is generally negligible compared to the number of formulas for instance. Furthermore, this method is qualified to be flexible since it permits to exploit efficiently all the existing propositional compilers. In addition, re-partitioning the stratified belief base by permuting some strata priority levels can be done efficiently without a re-compilation cost. However, we can not change the ranks of individual formulas unless we attach one variable per formula.

We have proposed a compilation of possibilistic logic knowledge bases that support weighted conclusions and possibilistic conditioning.

Our method can be favorably compared with other work concerned with compiling stratified belief bases. Note first that contrary to existing approaches, our method allows to compute weighted conclusion and possibilistic conditioning from an already compiled stratified belief base without a re-compilation cost. Let us mention the method proposed by

Cost-Marquis and Marquis in [Coste-Marquis and Marquis, 2004]. Like ours, it is flexible since it takes advantage of existing propositional knowledge bases compilation methods. On the other hand, this method is more general in the sense that it handles the two remaining preferred coherent subbases selection policies namely the lexicographic and inclusion-preference policies. However, the inherent cost is really expensive. In fact, they introduce a new propositional variable per formula. It is to note that this method was adapted in [Darwiche and Marquis, 2004] to compile weighted knowledge bases in the penalty logic framework.

This paper also brings several improvements regarding the compilation of possibilistic knowledge bases. In this context, a recent method has been introduced in [Benferhat and Prade, 2005]. At the encoding level, the latter uses additionally (k -1) binary clauses. Furthermore, it is based essentially on the use of forgetting variables and forgetting literals operations [Lang *et al.*, 2003] which reduces the number of target compilation languages that can be efficiently exploited. Indeed, this method exploits only the target compilation languages DNNF, DNF, PI and MODS which means that it is less flexible than the ours. Note first that contrary to existing approaches, our method allows to compute weighted conclusion and possibilistic conditioning from an already compiled stratified belief base without a re-compilation cost.

A future work is to extend these works for handling possibilistic conditioning with uncertain input [Dubois and Prade, 1997]. Another future work is to exploit results of this paper in order to provide an alternative implementation of existing propagation algorithms (such as junction trees algorithms) for possibilistic networks.

Acknowledgments: This work is supported by the national project ANR Blanc MICRAC (Modèles informatiques et cognitifs du raisonnement causal).

References

[Benferhat and Prade, 2005] S. Benferhat and H. Prade. Encoding formulas with partially constrained weights in possibilistic-like many-sorted propositional logic. In *Proceedings of IJCAI'05*, pages 1281–1286, 2005.

[Benferhat and Prade, 2006] S. Benferhat and H. Prade. Compiling possibilistic knowledge bases. In *Proceedings of ECAI'2006*, pages 337–341, Riva del Garda, Italy, 2006. IOS Press.

[Benferhat *et al.*, 1993] S. Benferhat, D. Dubois, C. Cayrol, J. Lang, and H. Prade. Inconsistency management and prioritized syntaxbased entailment. In *Proceedings of IJCAI'93*, pages 640–645, 1993.

[Brewka, 1989] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. In *Proceedings of IJCAI'89*, pages 1043–1048, 1989.

[Cadoli and Donini, 1997] M. Cadoli and F.M. Donini. A survey on knowledge compilation. *AI Communications*, 10:137–150, 1997.

[Coste-Marquis and Marquis, 2004] S. Coste-Marquis and P. Marquis. On stratified belief base compilation. *Annals of Mathematics and Artificial Intelligence*, 42(4):399–442, 2004.

[Darwiche and Marquis, 2002] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[Darwiche and Marquis, 2004] A. Darwiche and P. Marquis. Compiling propositional weighted bases. *Artificial Intelligence*, 157(1–2):81–113, 2004.

[Darwiche, 2001] A. Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

[Dubois and Prade, 1997] D. Dubois and H. Prade. A synthetic view of belief revision with uncertain inputs in the framework of possibility theory. *International Journal of Approximate Reasoning*, 17(2):295–324, 1997.

[Dubois *et al.*, 1994] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. *Handbook of Logic in Articial Intelligence and Logic Programming*, 3:439–513, 1994.

[Lang *et al.*, 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence - formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.

[Lang, 2000] J. Lang. Possibilistic logic: Complexity and algorithms. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, 5:179–220, 2000.

[Lehmann, 1995] D. Lehmann. Belief revision revisited. In *Proceedings of IJCAI'95*, pages 1534–1539, 1995.

[McCarthy, 1980] J. McCarthy. Circumscription : A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Nebel, 1994] B. Nebel. Base revision operations and schemes: semantics, representation and complexity. In *Proceedings of ECAI'94*, pages 341–345, 1994.

[Nebel, 1998] B. Nebel. How hard is it to revise a belief base? *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, 3:77–145, 1998.

[Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley Publishing Company, 1994.

[Rescher and Manor, 1970] N. Rescher and R. Manor. On inference from inconsistent premises. *Theory and Decision*, 1:179–217, 1970.

[Rescher, 1976] N. Rescher. *Plausible Reasoning*. Van Gorcum, Amsterdam, 1976.

[Selman and H.Kautz, 1996] B. Selman and H.Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[Williams, 1995] M.A. Williams. Iterated theory base change: a computational model. In *Proceedings of IJCAI'95*, pages 1541–1547, Montreal, Quebec, 1995.