Inférence sur des bases partiellement ordonnées: Implémentation

C. CAYROL
D. DUBOIS
F. TOUAZI

Rapport IRIT
RR - - 2015 - - 08 - -FR

Mai 2015

1 Introduction

Raisonner sur des bases de connaissances partiellement ordonnées est un problème qui a été largement étudié dans la littérature, et de nombreuses familles de logiques qui traitent cette problématique ont émergé de ces travaux. Notamment, les logiques conditionnelles [1, 2, 3], la logique non monotone [4] et des logiques pondérées telles que la logique possibiliste symbolique [5].

Dans ce rapport, nous proposons d'implémenter deux familles différentes de logiques. L'implémentation de telles logiques consiste à donner des réponses à certaines interrogations de la forme: est-ce qu'une formule ϕ est plus certaine (plausible ou préférée selon l'interprétation des formules de la base de connaissances) que ψ?

Dans [3, 6], nous avons proposé deux systèmes d'inférence : l'un (système $\mathcal{S}_1$) basé sur la relation de dominance faible le système, et l'autre ($\mathcal{S}_2$ son raffinement pré-additif). On a vu que le système $\mathcal{S}_1$ était correct et complet pour la sémantique de la certitude relative considérée dans plusieurs travaux [7, 8, 9, 5]. Nous avons montré des résultats de comparaison du systme $\mathcal{S}_1$ avec plusieurs travaux existants, notamment le système P, la logique des conditionnels, la logique modale et la logique possibiliste. Dans la suite, nous proposons une implémentation de ce système et la stratégie d'utilisation de ses règles d'inférence.

L'extension de la logique possibiliste dans le cas partiel en utilisant des poids symboliques [5] est aussi correcte et complète pour la même sémantique de certitude relative [10]. Une comparaison détaillée a été proposée entre ces approches qui a révélé un lien d'implication dans le sens o chaque conséquence d'une base partiellement ordonnée déduite en utilisant le système $\mathcal{S}_1$ est aussi une conséquence de la base possibiliste associée. Nous proposons d'implémenter cette logique en proposant deux méthodes d'inférence : une basée sur le calcul des sous-bases minimales inconsistantes et l'autre inspirée des concepts des ATMS.

Le rapport est structuré comme suit : la deuxième section est dédiée au problème SAT et son utilisation pour l'inférence classique. Dans la troisième section, on montre le processus et les étapes d'implémentation de système d'inférence $\mathcal{S}_1$ avec toutes les procédures nécessaires. Nous commenons par décrire les étapes et la stratégie utilisée pour faire de l'inférence en utilisant le système $\mathcal{S}_1$. La section quatre détaille l'implémentation des deux méthodes d'inférence en logique possibiliste symbolique.

2 Solveur SAT et inférence classique

Nous commençons par présenter un solveur SAT que l'on utilise par la suite pour répondre à des questions en logique classique.

2.1 Solveur SAT

Un solveur SAT est un programme qui décide si une formule de logique propositionnelle sous forme causale est satisfiable (ou consistante) ou non [11]. Un problème SAT est un problème de décision dont le but est de savoir s'il existe une solution à une série d'équations logiques données

(ensemble de clauses propositionnelles). Résoudre un problème SAT revient donc à décider s'il existe, ou non, une configuration d'affectations "Vrai/Faux" à des variables d'entrée qui permette de satisfaire une expression logique sous la forme conjonctive (dans le cas satisfiable, le solveur SAT fournit une interprétation qui satisfait toutes les clauses).

Définition 1 *Soit B une base de clauses propositionnelles. $SAT(B)$ est résolu positivement (i.e. admet pour réponse oui) si et seulement si B est satisfiable.*

Un solveur SAT ne prend en entrée que des clauses. Cependant, dans un langage propositionnel, les formules propositionnelles sont quelconques. Donc avant de générer le problème SAT associé à une formule propositionnelle, il faut transformer cette formule en clauses. Il existe dans la littérature des méthodes efficaces pour transformer une formule propositionnelle quelconque sous forme clausale [12].

Un solveur SAT nous permet donc de tester la consistance ou l'inconsistance d'une base propositionnelle. Etant donné une base propositionnelle B, on considère la conjonction de toutes les clauses issues de la base B.

Exemple 1

- *Si on considère la base $B = \{\neg x \vee y, x\}$, en générant le problème SAT associé, on trouve que SAT(B) est résolu positivement donc B est consistante.*
- *Si on considère la base $B = \{x, \neg x \vee y, x \wedge y, \neg x, \neg y\}$. On transforme la base en forme clausale $B' = \{x, \neg x \vee y, y, \neg x, \neg y\}$. SAT($B'$) est résolu négativement donc B est inconsistante.*

2.2 Problème d'inférence

Comme on a vu dans la sous-section précédente, un solveur SAT répond essentiellement à des questions de satisfiabilité ou non-satisfiabilité d'un ensemble de clauses propositionnelles.

Dans notre implémentation, nous allons utiliser un solveur SAT pour répondre à des questions de la forme $B \vdash \phi$ où B est une base logique composée de formules propositionnelles et ϕ est une formule propositionnelle.

Étant donné une base B, répondre à la question $B \vdash \phi$ revient à construire un problème SAT pour la base $B \cup \{\neg\phi\}$ et établir que ce problème est résolu négativement.

Exemple 1 (suite) On considère la base $B = \{\neg x \vee y, x\}$. On cherche à montrer que $B \vdash y$. $B' = B \cup \{\neg y\} = \{\neg x \vee y, x, \neg y\}$. SAT($B'$) est résolu négativement donc $B \vdash y$.

3 Implémentation du système d'inférence $\mathcal{S}_1$

Dans [3], nous avons présenté deux systèmes d'inférence $\mathcal{S}_1$ et $\mathcal{S}_2$. Dans cette section, on détaille le processus d'inférence du système $\mathcal{S}_1$ ainsi que les procédures à utiliser. Avant de commencer, nous rappelons le langage utilisé.

3.1 Rappel sur la syntaxe

Nous considérons un langage propositionnel classique $\mathcal{L}$. On note $\phi,\ \psi, \cdots$ les formules propositionnelles construites à l'aide des connecteurs usuels $\neg, \wedge, \vee$ de la logique classique et des atomes. La syntaxe proposée consiste à encapsuler le langage $\mathcal{L}$ à l'intérieur d'un langage équipé d'un connecteur binaire $>$ interprété comme une relation d'ordre $>$. Formellement, un littéral Φ de $\mathcal{L}_>$ est de la forme $\phi > \psi$ ou $\neg(\phi > \psi)$ avec ϕ et ψ des formules propositionnelles classiques de $\mathcal{L}$.

Une formule de $\mathcal{L}_>$ est donc soit un littéral Φ de $\mathcal{L}_>$, soit de la forme $\Psi \wedge \Phi$ avec Ψ, Φ des formules de $\mathcal{L}_>$. Une base $\mathcal{B}$ est un sous-ensemble fini de $\mathcal{L}_>$. Dans la suite on ne raisonnera qu'à partir de bases ne contenant que des littéraux positifs de la forme $\phi > \psi$ (même si la fermeture déductive de $\mathcal{B}$ pourra contenir des littéraux négatifs). Une telle formule peut être interprétée comme: un agent a plus de certitude dans ϕ que dans ψ.

On associe à une base partiellement ordonnée $(\mathcal{K}, >)$ un ensemble de formules de la forme $\phi > \psi$ formant une base $\mathcal{B}_\mathcal{K}$ de $\mathcal{L}_>$. Notons que si $\phi > \psi$ et $\psi > \xi$ sont dans $\mathcal{B}_\mathcal{K}$, alors $\phi > \xi \in \mathcal{B}_\mathcal{K}$ puisqu'on a un ordre partiel sur $\mathcal{K}$. Remarquons que pour retrouver $(\mathcal{K}, >)$ à partir de $\mathcal{B}_\mathcal{K}$, il faut que chaque formule de $\mathcal{K}$ apparaisse dans une formule de $\mathcal{B}_\mathcal{K}$ (il n'y a pas de formule incomparable avec toutes les autres dans $(\mathcal{K}, >)$).

Dans la suite, pour chaque base partiellement ordonnée $(\mathcal{K}, >)$, $(\mathcal{K}, >) \vdash_\mathcal{S} \Phi$ dénote que Φ est une conséquence de la base $\mathcal{B}_\mathcal{K}$ dans le système d'inférence $\mathcal{S}$.

3.2 Processus d'inférence avec le système $\mathcal{S}_1$

Le système $\mathcal{S}_1$ est un système d'inférence syntaxique constitué de 2 axiomes et 3 règles d'inférences. Nous avons déjà montré qu'il est correct et complet pour la sémantique de la certitude relative basée sur la relation $\succ_{dofs}$ [3]:

Axiomes

ax_1 : $\top > \bot$

ax_2 : Si $\psi \vDash \phi$ alors $\neg(\psi > \phi)$

Règles primitives

RI_1 : Si $\chi > \phi \wedge \psi$ et $\psi > \phi \wedge \chi$ alors $\psi \wedge \chi > \phi$ (Q^d)

RI_2 : Si $\phi > \psi$, $\phi \vDash \phi'$ et $\psi' \vDash \psi$ alors $\phi' > \psi'$ (POI)

RI_3 : Si $\phi > \psi$ alors $\neg(\psi > \phi)$ (AS)

Règles dérivées

RI_4 : Si $\phi > \psi$ et $\psi > \chi$ alors $\phi > \chi$ (T).

RI_5 : Si $\psi > \phi$ et $\chi > \phi$ alors $\psi \wedge \chi > \phi$ (ADJonction)

RI_6 : Si $\phi \to \chi > \phi \to \neg\chi$ et $\psi \to \chi > \psi \to \neg\chi$ alors $(\phi \vee \psi) \to \chi > (\phi \vee \psi) \to \neg\chi$ (OR^d)

RI_7 : Si $\chi \to \phi > \chi \to \neg\phi$ et $\chi \to \psi > \chi \to \neg\psi$ alors $\chi \to (\phi \wedge \psi) > \chi \to \neg(\phi \wedge \psi)$ (CCC^d)

RI_8 : Si $\phi \to \psi > \phi \to \neg\psi$ et $(\phi \wedge \psi) \to \chi > (\phi \wedge \psi) \to \neg\chi$ alors $\phi \to \chi > \phi \to \neg\chi$ (CUT^d)

RI_9 : Si $\phi \to \psi > \phi \to \neg\psi$ et $\phi \to \chi > \phi \to \neg\chi$ alors $(\phi \wedge \psi) \to \chi > (\phi \wedge \psi) \to \neg\chi$ (MF^d)

L'implémentation de ce système consiste à utiliser chaque règle d'inférence primitive. Cependant, l'utilisation des règles d'inférences nécessite une stratégie.

Étant donné une base partiellement ordonnée, on considère une Φ formule de notre langage $\mathcal{L}_>$ de la forme $\phi > \psi$. Le système $\mathcal{S}_1$ cherche à répondre à des questions du genre $\phi > \psi$? en utilisant les règles d'inférences décrites ci-dessus, à partir de la base $\mathcal{B}_\mathcal{K}$.

Il est clair que la seule règle d'inférence qui nous permet de déduire des littéraux comportant des nouvelles formules du langage $\mathcal{L}$ est la règle d'inférence RI_2.

On note $\Phi \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$ la possibilité de déduire Φ en utilisant les règles d'inférence RI_1, RI_2 et RI_3 du système $\mathcal{S}_1$ sur la base $\mathcal{B}_\mathcal{K}$. Ce processus n'est utile que dans le cas où la formule Φ n'est pas présente dans la base initiale.

Pour déduire la formule Φ avec le système $\mathcal{S}_1$ on distingue deux cas:

1. La formule à déduire Φ contient une conjonction dans la partie gauche. Dans ce cas, seule la règle RI_1 permet de déduire cette formule puisque la règle RI_2 ne peut pas affaiblir la partie gauche d'une formule de la forme $\phi > \psi$. La stratégie consiste à vérifier les hypothèses d'une possible application de la règle d'inférence RI_1 qui nous donne comme résultat la formule initiale. Soit la formule à déduire $\phi \wedge \chi > \psi$, il suffit de vérifier que $\phi > \psi \wedge \chi \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$ et $\psi > \phi \wedge \chi \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$.

2. La formule à déduire Φ ne contient pas de conjonction dans la partie gauche. Dans ce cas, seule la règle RI_2 nous permet de déduire cette formule (on admet toujours que cette formule n'est pas déjà dans la base).

   Notre stratégie d'utilisation des règles d'inférence consiste à fermer la base par les règles d'inférence RI_1, RI_3 pour générer $\mathcal{C}_{1,3}(\mathcal{B}_\mathcal{K})$, un fragment de la fermeture déductive partiellement ordonnée. A nouveau, on distingue deux cas:

   (a) RI_2 permet de déduire la formule Φ à partir de la fermeture intermédiaire $\mathcal{C}_{1,3}(\mathcal{B}_\mathcal{K})$.

   (b) RI_2 ne permet pas de déduire la formule. Dans ce cas,

   - On cherche une formule Υ de la forme $\xi > \psi$ et une autre formule Ξ de la forme $\chi > \xi$ telle que $\chi \wedge \xi \vdash \phi$. L'idée est de pouvoir appliquer la règle d'inférence RI_1 pour obtenir une formule de la forme $\xi \wedge \chi > \psi$.
   - On vérifie que $\xi > \psi \wedge \chi \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$ et $\chi > \psi \wedge \xi \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$,
   - Par la règle RI_2, puisque $\xi \wedge \chi \vdash \phi$ et $\psi \vdash \psi$, on déduit $\phi > \psi$.

On essaye avec tous les couples de formules possibles. En cas d'échec la formule Φ n'est pas déductible dans le système $\mathcal{S}_1$.

Notons qu'à chaque utilisation des règles d'inférence RI_1, RI_2 on applique la règle RI_3. Cette règle sert essentiellement à détecter l'inconsistance dans la base.

Exemple 2 *Soit* $(\mathcal{K}, >) = \{x > \neg x \vee y, \neg x \vee y > \neg x, \neg x \vee y > \neg y\}$ *une base partiellement ordonnée.*
On veut savoir si $y > \neg y$ *se déduit de* $(\mathcal{K}, >)$ *par* $\mathcal{S}_1$.
Puisque il n'y a pas de conjonction dans la partie gauche, on est dans le second cas. Par RI_2 *on ne peut rien déduire.*
On est donc dans le cas (2b). On considère les deux formules:

- $\Upsilon = \neg x \vee y > \neg y$ *car elle contient* $\neg y$ *dans sa partie droite;*
- $\Xi = x > \neg x \vee y$, *puisque* $(\neg x \vee y) \wedge x \vdash y$

On vérifie que:

- $(\neg x \vee y) > \neg y \wedge x \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$ *par* RI_2 *sur la fermeture* $\mathcal{C}_{1,3}(\mathcal{B}_\mathcal{K})$
- $x > (\neg x \vee y) \wedge \neg y \in \mathcal{C}_{1,2,3}(\mathcal{B}_\mathcal{K})$ *par* RI_2 *sur la fermeture* $\mathcal{C}_{1,3}(\mathcal{B}_\mathcal{K})$

Donc on déduit $x \wedge (\neg x \vee y) > \neg y$ *et par* RI_2 *que* $(\mathcal{K}, >) \vdash_{\mathcal{S}_1} y > \neg y$.

L'étape (2b) consiste en fait à appliquer les règles d'inférence dérivées (i.e. à refaire les preuves des règles dérivées à partir des deux règles RI_1, RI_3). Nous proposons donc de remplacer cette étape par la fermeture de la base par les règles d'inférence $RI_1, RI_3, RI_4, RI_5, RI_6, RI_7, RI_8, RI_9$ que l'on note par $\mathcal{C}_{\{1,3,4,5,6,7,8,9\}}(\mathcal{B}_\mathcal{K})$, suivie de l'application de la règle RI_2 sur cette fermeture partielle.

L'ordre d'application des règles est le suivant : $RI_4, RI_5, RI_1, RI_6, RI_7, RI_8, RI_9$. Il est justifié par le fait que les règles RI_4, RI_5 produisent des conséquences très utiles. On génère ensuite les conséquences possibles par RI_1 (on commence par RI_4, RI_5 pour générer autant de conséquences possibles pour cette règle) et en dernière position, les autres règles qui sont moins essentielles. Notons qu'à chaque utilisation d'une règle d'inférence, on vérifie la possibilité d'appliquer les règles précédentes, par exemple si on produit une conséquence avec la règle RI_1 alors on considère les deux règles RI_4, RI_5. La règle RI_3 sert essentiellement à détecter l'inconsistance dans la base, donc elle sera appliquée à la fin.

Le processus d'inférence est résumé dans l'algorithme 1. Il prend en entrée une base partiellement ordonnée et une formule à déduire. Il donne en sortie une réponse Vrai (True) ou Faux (False) pour la déductibilité de la formule à partir de la base $\mathcal{B}_\mathcal{K}$:

Algorithm 1 $inf\_Sys\_1(\mathcal{B}_\mathcal{K}, \phi > \psi)$

if $\phi > \psi \in \mathcal{B}_\mathcal{K}$ **then**
  return True; ▷ le cas où la conséquence existe dans la base
end if
if $\phi = \varphi \wedge \chi$ **then** ▷ Point 1, la conséquence contient une conjonction à gauche
  return $inf\_Sys\_1(\mathcal{B}_\mathcal{K}, \varphi > \chi \wedge \psi) \wedge inf\_Sys\_1(\mathcal{B}_\mathcal{K}, \chi > \psi \wedge \varphi)$**;**
else
  $\mathcal{B}_\mathcal{K} := \mathcal{C}_{1,3,4,5,6,7,8,9}(\mathcal{B}_\mathcal{K})$;
  for $\Phi \in (\mathcal{K}, >)$ **do**
    if $\phi > \psi \in \mathcal{C}_2(\Phi)$ **then**
      return True; ▷ Point 2
    end if
  end for
end if
return False;

3.3 Implémentation des règles d'inférence

L'implémentation de chaque règle d'inférence consiste à vérifier les hypothèses de cette dernière et générer la conséquence si toutes les hypothèses sont vraies.

Nous pouvons voir que la règle d'inférence RI_2 est particulière puisque dans ses hypothèses elle fait appel à la déduction classique. Donc pour utiliser cette règle d'inférence nous devons faire appel à un solveur SAT classique.

3.3.1 Implémentation des règles d'inférence primitives

L'implémentation des 3 règles d'inférences primitives est résumée dans l'algorithme 2.

Algorithm 2 Règles d'inférence RI_1, RI_2, RI_3

procedure $RI_1(\psi > \phi \wedge \chi, \chi > \phi \wedge \psi)$ ▷ Règle d'inférence RI_1
    $\Phi := \psi \wedge \chi > \phi$
return Φ
end procedure

procedure $RI_2(\phi > \psi, \phi' > \psi')$ ▷ Règle d'inférence RI_2
    On génère le problème SAT $\phi \vDash \phi'$ $SAT(\{\phi \wedge \neg\phi'\})$
    On génère le problème SAT $\psi' \vDash \psi$ $SAT(\{\psi' \wedge \neg\psi\})$
    if not($SAT(\{\phi \wedge \neg\phi'\})$) and not($SAT(\{\psi' \wedge \neg\psi\})$) **then**
        $\Phi := \phi' > \psi'$
return Φ
    end if
end procedure

procedure $RI_3(\phi > \psi)$ ▷ Règle d'inférence RI_3
    $\Phi := \neg(\psi > \phi)$
return Φ
end procedure

3.3.2 Règles d'inférence dérivées

L'implémentation des règles d'inférence dérivées est résumée dans l'algorithme 3.

Algorithm 3 Règles d'inférence dérivées

procedure $RI_4(\Psi = \phi > \psi, \Xi = \chi > \xi)$ ▷ Règle d'inférence RI_4
  if $\psi = \chi$ **then**
    $\Phi = \phi > \xi$
  else
    if $\xi = \phi$ **then**
      $\Phi := \chi > \psi$
return Φ
    end if
  end if
end procedure

procedure $RI_5((\Psi, \Xi))$ ▷ Règle dérivée RI_5
  if $\Psi = \phi > \chi$ **and** $\Xi = \psi > \chi$ **then**
    $\Phi := \phi \wedge \psi > \chi$
return Φ
  end if
end procedure

procedure $RI_6((\Psi, \Xi))$ ▷ Règle dérivée RI_6
  if $\Psi = \phi \to \chi > \phi \to \neg\chi$ **and** $\Xi = \psi \to \chi > \psi \to \neg\chi$ **then**
    $\Phi := (\phi \vee \psi) \to \chi > (\phi \vee \psi) \to \neg\chi$
return Φ
  end if
end procedure

procedure $RI_7((\Psi, \Xi))$ ▷ Règle dérivée RI_7
  if $\Psi = \phi \to \chi > \phi \to \neg\chi$ **and** $\Xi = \phi \to \psi > \phi \to \neg\psi$ **then**
    $\Phi := \phi \to (\psi \wedge \chi) > \phi \to \neg(\psi \wedge \chi)$
return Φ
  end if
end procedure

procedure $RI_8((\Psi, \Xi))$ ▷ Règle dérivée RI_8

  if $\Psi = \phi \rightarrow \chi > \phi \rightarrow \neg\chi$ **and** $\Xi = (\phi \wedge \psi) \rightarrow \chi > (\phi \wedge \psi) \rightarrow \neg\chi$ **then**

    $\Phi := \phi \rightarrow \chi > \phi \rightarrow \neg\chi$ **return** Φ

  end if

end procedure

procedure $RI_9((\Psi, \Xi))$ ▷ Règle dérivée RI_9

  if $\Psi = \phi \rightarrow \chi > \phi \rightarrow \neg\chi$ **and** $\Xi = \phi \rightarrow \psi > \phi \rightarrow \neg\psi$ **then**

    $\Phi := (\phi \wedge \psi) \rightarrow \chi > (\phi \wedge \psi) \rightarrow \neg\chi$ **return** Φ

  end if

end procedure

Exemple 2 (suite) $(\mathcal{K}, >) = \{x > \neg x \vee y, \neg x \vee y > \neg x, \neg x \vee y > \neg y\}$.
On veut savoir si $y > \neg y$ se déduit de $(\mathcal{K}, >)$ par $\mathcal{S}_1$.
On ferme d'abord la base avec les règles d'inférence autres que RI_2, on obtient donc $\mathcal{C}_{\{1,3,4,5\}}(\mathcal{B}_\mathcal{K})$:

$$ParRI_4 : \quad x > \neg x; \tag{1}$$

$$ParRI_4 : \quad x > \neg y; \tag{2}$$

$$ParRI_5 : \ x \wedge (\neg x \vee y) > \neg y; \tag{3}$$

$$ParRI_5 : x \wedge (\neg x \vee y) > \neg x; \tag{4}$$

Ensuite en utilisant la règle d'inférence RI_2 sur la formule $x \wedge \neg x \vee y > \neg y$ (3) on trouve: $y > \neg y \in \mathcal{C}_{1,2,3,4,5,6,7,8,9}(\mathcal{B}_\mathcal{K})$.

3.4 Implémentation du système d'inférence par les coupes

Dans cette partie nous présentons l'implémentation de l'inférence par les coupes. Nous rappelons d'abord les principes de cette inférence.

3.4.1 Rappel sur l'inférence avec les coupes

Une autre méthode d'inférence de la logique possibiliste a été proposée dans [3]. L'idée est d'utiliser l'inférence classique sur une partie de la base partiellement ordonnée calculée en utilisant les coupes.

Définition 2 (Inférence par les coupes) *Soit $\psi \in \mathcal{K}$, on définit $\mathcal{K}_\psi^> = \{\gamma | \gamma \in \mathcal{K}$ et $\gamma > \psi\}$. La formule $\phi > \psi$ se déduit par les coupes d'une base partiellement ordonnée $(\mathcal{K}, >)$, ce qu'on note $(\mathcal{K}, >) \vdash_c \phi > \psi$, ssi $\mathcal{K}_\psi^>$ est consistante et $\mathcal{K}_\psi^> \vdash \phi$.*

La fermeture déductive par les coupes d'une base partiellement ordonnée $\mathcal{C}_c^+(\mathcal{K}, >)$ est définie par:

$$\mathcal{C}_c^+(\mathcal{K}, >) = \{\phi > \psi \in \mathcal{L}_> : (\mathcal{K}, >) \vdash_c \phi > \psi\}.$$

Nous avons vu que l'inférence par les coupes était une inférence correcte mais trop faible pour calculer toute la fermeture déductive du système $\mathcal{S}_1$.

Proposition 1 *[10] Soit $(\mathcal{K}, >)$ une base partiellement ordonnée. Pour toute formule ψ de $\mathcal{K}$, si $(\mathcal{K}, >) \vdash_c \phi > \psi$ alors $(\mathcal{K}, >) \vdash_{\mathcal{S}_1} \phi > \psi$. La réciproque est fausse.*

3.4.2 Implémentation de l'inférence avec les coupes

Cette inférence est basée sur l'inférence classique. Étant donné une base partiellement ordonnée $(\mathcal{K}, >)$ et deux formules ϕ, ψ, pour montrer que $\phi > \psi$, la première étape consiste à calculer la coupe au niveau ψ c'est à dire toutes les formules dans $\mathcal{K}$ qui dominent ψ. Ensuite, par déduction classique, on vérifie si les formules qui appartiennent à la coupe permettent d'inférer la formule ϕ. Notons que si $\psi \notin \mathcal{K}$, il est impossible de déduire des conséquences de la forme $\phi > \psi$. L'algorithme 4 résume cette procédure.

Algorithm 4 Inf_Coup($\mathcal{B}_\mathcal{K}, \phi > \psi$)

1: **if** $\psi \notin \mathcal{K}$ **then**
2: return False;
3: **else**
4: $\mathcal{K}_\psi^> := \{\gamma | \gamma > \psi \in \mathcal{B}_\mathcal{K}\}$;
5: On génère le problème $SAT(\mathcal{K}_\psi^> \cup \{\neg\phi\})$;
6: **return** $not(SAT(\mathcal{K}_\psi^> \cup \{\neg\phi\}))$;
7: **end if**

Nous avons montré que l'inférence par les coupes est correcte, donc tout ce qu'on déduit avec cette inférence est une conséquence du système d'inférence $\mathcal{S}_1$. Il est clair que l'inférence par les coupes nécessite peu de calculs, donc on pourra l'intégrer dans le processus d'inférence du système $\mathcal{S}_1$. Étant donné une base partiellement ordonnée $(\mathcal{K}, >)$ et deux formules ϕ, ψ, on commence par tester la déductibilité par l'inférence par les coupes, puis en cas d'échec on procède par l'inférence dans $\mathcal{S}_1$.

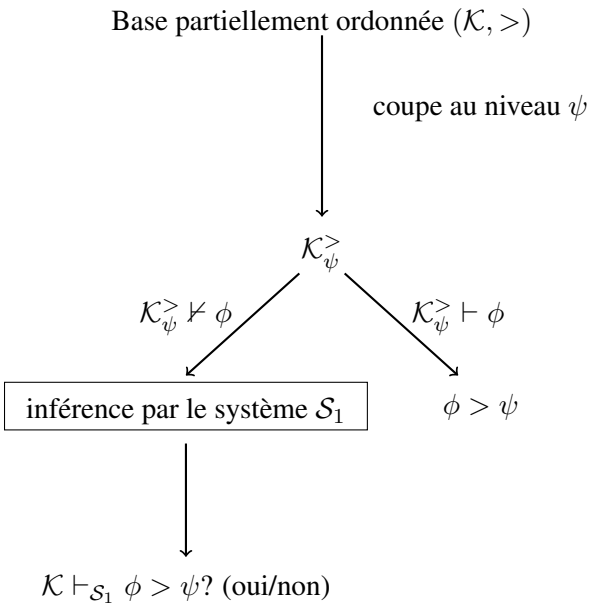

Figure 1: schéma explicatif du processus d'inférence du système $\mathcal{S}_1$ fusionné avec le système d'inférence par les coupes

4 Implémentation du système d'inférence possibiliste

Nous avons vu que le système d'inférence possibiliste avec des poids symboliques est plus puissant que le système d'inférence $\mathcal{S}_1$, puisque toute conséquence du système $\mathcal{S}_1$ est une conséquence du système d'inférence possibiliste symbolique [10]. En effet, ce dernier utilise le principe de spé cificité minimale pour compléter l'ordre partiel. Dans cette section, nous allons présenter deux méthodes d'inférence syntaxique qui calculent le degré de nécessité d'une formule possibiliste. La première méthode est basée sur l'utilisation de la notion de sous-base inconsistante minimale. La seconde est inspirée du raisonnement abductif. Nous supposons que les poids portant sur des formules de la base possibiliste sont élémentaires, avec possibilité d'attribuer le même poids à différentes formules.

4.1 Rappel sur la logique possibiliste symbolique (LPosS)

Dans cette sous-section, nous considérons un langage propositionnel où les formules sont notées ϕ_1, ..., ϕ_n et Ω est l'ensemble des interprétations.
On manipule des formules pondérées (ϕ_j, p_j) où ϕ_j est une formule propositionnelle et $p_j \in]0, 1]$. (ϕ_j, p_j) s'interprète par $N(\phi_j) \geq p_j$, avec N une mesure de nécessité [9]. p_j est alors vu comme un degré de certitude.

Une base possibiliste est un ensemble de formules pondérées $\Sigma = \{(\phi_j, p_j) \mid j = 1, \ldots, m\}$.
Soit Σ^* la base logique classique associée avec la base possibiliste Σ:

$$\Sigma^* = \{\phi : \exists p > 0, (\phi, p) \in \Sigma\}$$

Comme les poids ne sont que des bornes inférieures ils n'ajoutent jamais d'inconsistance à la base. La seule cause d'inconsistance provient de l'inconsistance au sens classique de $\Sigma^* = \{\phi_1, \ldots, \phi_m\}$.

En logique possibiliste, toute base possibiliste est caractérisée par un degré d'inconsistance $Inc(\Sigma)$. Si la base Σ^* est inconsistante, $Inc(\Sigma) = 1 - \max_\omega \pi_\Sigma(\omega)$ représente le degré d'inconsistance de la base Σ. Ce degré d'inconsistance peut-être aussi calculé par la formule suivante:

$$Inc(\Sigma) = \max\{p | \Sigma \vdash_\pi (\bot, p)\}$$

où $\vdash_\pi$ désigne l'inférence syntaxique en logique possibiliste.

On considère maintenant que l'on ne dispose que de connaissances partielles sur l'ordre entre les poids des formules. On utilise alors des poids symboliques et des contraintes sur ces poids symboliques. L'ensemble des poids symboliques est obtenu à l'aide d'un ensemble fini H de variables $a_1, \ldots, a_k, \ldots$ sur $]0, 1]$ et de $\max$ / $\min$ expressions construites sur H. Les éléments de H sont appelés poids symboliques élémentaires. Une formule (ϕ_i, p_i) est toujours interprétée comme $N(\phi_i) \geq p_i$.

La connaissance sur l'ordre entre les poids symboliques est codée par un ensemble $\mathcal{C} = \{p_j > p_j,\ j = 1, \cdots, s\}$ de contraintes strictes sur ces poids. Une base possibiliste symbolique est composée de deux parties $(\Sigma\ ,\ \mathcal{C})$: Σ est un ensemble de formules possibilistes, $\mathcal{C}$ est un ensemble de contraintes sur les poids symboliques. Ces contraintes peuvent être vues comme une base partiellement ordonnée. Il est clair que seule la clàŽture transitive est considérée (une conjonction entre deux poids n'a pas de sens dans ce contexte). Donc, le système d'inférence $\mathcal{S}_1$ restreint à la règle d'inférence $RI4$ pourra être utilisé pour faire des déductions sur les poids symboliques (la fermeture transitive de la base partiellement ordonnée $\mathcal{C}$).

4.2 Vers une méthode d'inférence en logique possibiliste symbolique

L'inférence possibiliste symbolique est correcte et complète pour la sémantique de la certitude relative [10]. Il suffit donc de calculer l'un des degrés de nécessité. On a choisi de calculer le degré syntaxique $N_\Sigma^\vdash$ défini comme suit:

$$N_\Sigma^\vdash(\phi) = \max_{B \subseteq \Sigma^*, B \vdash \phi} \min_{\phi_j \in B} p_j. \tag{5}$$

Pour des raisons de simplification, dans la suite de ce rapport, ce degré de nécessité syntaxique sera noté $N_\Sigma(\phi)$.

Calculer le degré de nécessité (syntaxique) d'une formule ϕ revient à trouver tous les sous-bases minimales pour l'inclusion qui infèrent ϕ (voir équation (5)).
Certaines des sous-bases minimales qui impliquent ϕ peuvent être inconsistantes. Dans ce cas, elles sont minimales inconsistantes dans Σ^*.

Lemme 1 *Soit $B \subseteq \Sigma^*$ inconsistante et minimale qui implique ϕ. Alors B est minimale inconsistante dans Σ^*.*

Preuve du Lemme 1: Soit $B \subseteq \Sigma^*$ où B est inconsistante et minimale qui implique ϕ. Supposons que $B' \subset B$ est minimale inconsistante. Alors $B' \vdash \phi$ ce qui contredit l'hypothèse sur B. □

Donc si $B \subseteq \Sigma^*$ est une sous-base minimale qui implique ϕ, soit B est une sous-base consistante, soit B est inconsistante minimale dans Σ^*. Cependant, il peut arriver que certaines sous-bases minimales inconsistantes dans Σ^* ne soient pas des sous-bases minimales qui impliquent ϕ.

Proposition 2 *Soit $B_1, \cdots, B_k$ les sous-bases minimales consistantes dans Σ^* qui impliquent ϕ. Soit $I_1, \cdots, I_l$ les sous-bases minimales inconsistantes dans Σ^* qui ne contiennent aucune base B_j. On a :*

$$N_\Sigma(\phi) = \max(\max_{i=1}^{k} \min_{\phi_j \in B_i} p_j, \max_{i=1}^{l} \min_{\phi_j \in I_i} p_j)$$

Notons que $B \subseteq \Sigma^*$ est minimale impliquant ϕ ssi B minimale telle que $B \cup \{\neg\phi\}$ est inconsistante.

Proposition 3 *Soit $(\Sigma, \mathcal{C})$ une base possibiliste symbolique et B une sous-base de Σ^*.*

1. *Si K est une sous-base inconsistante minimale de $\Sigma^* \cup \{\neg\phi\}$ qui contient $\neg\phi$, alors $K \setminus \{\neg\phi\}$ est minimale consistante impliquant ϕ.*
2. *Si B est minimale consistante impliquant ϕ alors $B \cup \{\neg\phi\}$ est une sous-base inconsistante minimale de $\Sigma^* \cup \{\neg\phi\}$.*

Preuve de la Proposition 3:
1) Soit $K \subseteq \Sigma^* \cup \{\neg\phi\}$ telle que $\neg\phi \in K$ et K est minimale inconsistante dans $\Sigma^* \cup \{\neg\phi\}$. Soit $B = K \setminus \{\neg\phi\}$ où $B \subseteq \Sigma^*$. B est consistante, puisque K est minimale inconsistante, et $B \vdash \phi$ puisque $B \cup \{\neg\phi\}$ est inconsistante. Supposons que B n'est pas minimale qui implique ϕ, il existe $B' \subset B$ tel que $B' \vdash \phi$. Donc $B' \cup \{\neg\phi\}$ est inconsistante et $B' \cup \{\neg\phi\} \subset K$ contradiction avec l'hypothèse sur K.
2) Soit $B \subseteq \Sigma^*$ minimale qui implique ϕ et supposons que B est consistante. Soit $K = B \cup \{\neg\phi\}$, K est inconsistante dans $\Sigma^* \cup \{\neg\phi\}$ et contient $\neg\phi$. Supposons que K n'est pas minimale inconsistante dans $\Sigma^* \cup \{\neg\phi\}$. Il existe $K' \subset K$ telle que K' est inconsistante, nous pouvons même supposer que K' est minimale inconsistante.

- Soit $\neg\phi \in K'$, donc par le premier point, on a $K' \setminus \{\neg\phi\}$ est minimale qui implique ϕ et $K' \setminus \{\neg\phi\} \subset K \setminus \{\neg\phi\} = B$ contradiction avec l'hypothèse sur B.
- Soit $\neg\phi \notin K'$, alors $K' \subset B$, puisque K' est inconsistante, B est également inconsistante, ce qui contredit l'hypothèse sur B.

Alors K est inconsistante dans $\Sigma^* \cup \{\neg\phi\}$.

□

Par la proposition 2 et la proposition 3, afin de calculer $N(\phi)$, nous devons déterminer:

- $\{B \subseteq \Sigma^* | B \cup \{\neg\phi\}$ est une sous-base minimale inconsistante dans $\Sigma^* \cup \{\neg\phi\}\}$
- Les sous-bases minimales inconsistantes dans Σ^* qui ne contiennent aucune des B_i dans le premier ensemble.

Le calcul ci-dessus revient à un problème bien connu dans la littérature qui consiste à déterminer l'ensemble des sous-bases minimales inconsistantes d'un ensemble S, noté par $MIS(S)$. Notons

$$\mathcal{B}^{\vdash}(\phi) = \{B \subseteq \Sigma^* | B \cup \{\neg\phi\} \in MIS(\Sigma^* \cup \{\neg\phi\})\}$$

et

$$\mathcal{B}_i(\phi) = \{B \in MIS(\Sigma^*) | B\text{ne contient aucune base de}\mathcal{B}^{\vdash}(\phi)\}.$$

Soit $\mathcal{B}(\phi) = \mathcal{B}^{\vdash}(\phi) \bigcup \mathcal{B}_i(\phi)$. Alors le degré de nécessité d'une formule ϕ peut être calculé par :

$$N_{\Sigma}(\phi) = \max_{B_i \in \mathcal{B}(\phi)} \min_{\phi_j \in B_i} p_j \tag{6}$$

Dans la suite, nous proposons plusieurs procédures qui nous permettront de calculer l'ensemble $MIS(S)$.

4.3 Calcul des sous-bases minimales inconsistantes

Dans la littérature, il existe plusieurs stratégies possibles pour calculer les sous-bases minimales inconsistantes d'une base propositionnelle. Dans la suite, nous rappelons ces stratégies et nous choisissons la plus efficace. Notons que ces stratégies portent sur des bases propositionnelles sous forme clausale.

4.3.1 Stratégie Bottom-up

Cette stratégie (proposée dans [13] dans un contexte différent), consiste à commencer la génération par des sous-bases inconsistantes qui contiennent au moins deux clauses.

Étant donné une base propositionnelle S, on génère toutes les bases possibles B_i qui contiennent exactement deux clauses. Si une base B_i est inconsistante alors on la rajoute à l'ensemble des sous-bases minimales inconsistantes, sinon à chaque itération on rajoute aux bases consistantes B_i une nouvelle formule tout en assurant que les nouvelles bases restent minimales, puis on refait les tests jusqu'à ce que l'on trouve des bases inconsistantes.

Pour éviter de faire des calculs inutiles, on commence par tester si la base S est inconsistante pour être certain qu'il y ait au moins une sous-base inconsistante. L'algorithme 5 calcule les sous-bases minimales inconsistantes d'un ensemble S via la stratégie Bottom-up.

Algorithm 5 MI_B(S)

$MIS = \emptyset$;
if $SAT(S)$ **then**
    return MIS;
else
    $\mathcal{B} = \{\{\phi_i, \phi_j\}, \cdots, \{\phi_n, \phi_m\}\}$;
    $MIS := Bottom\_up(\mathcal{B}, S, \emptyset)$;
end if
return MIS;

L'algorithme 6 calcule à chaque itération des sous-bases minimales inconsistantes. Il prend en entrée $\mathcal{B}$ l'ensemble des sous-bases candidates (au départ, chaque sous-base contient deux clauses, puis à chaque itération on rajoute une clause à chaque sous-base), S l'ensemble des clauses et MIS l'ensemble des sous-bases minimales inconsistantes déjà calculées.

Algorithm 6 Bottom_up($\mathcal{B}, S, MIS$))

if $\mathcal{B} = \emptyset$ **then**
return $\emptyset$;
else
    $\mathcal{B}_{next} := \emptyset$;
    for $B_j \in \mathcal{B}$ **do**
        if $not(SAT(B_j))$ **then**
            add B_j to MIS;
        else
            add B_j to $\mathcal{B}_{next}$; ▷ Stocker les bases $\mathcal{B}_j$ consistantes pour la suite
        end if
    end for
    for $B_j \in \mathcal{B}_{next}$ **do** ▷ Pour chaque base consistante on rajoute à cette base une formule
        for $\phi_i \in S$, and $\phi_i \notin B_j$ **do** ▷ Choisir une formule ϕ_i dans S
            if $B_j \cup \{\phi_i\}$ ne contient pas une sous-base de MIS **then**
                remplacer B_j par $B_j \cup \{\phi_i\}$ dans $\mathcal{B}_{next}$;
            end if
        end for
    end for
return Bottom_up($\mathcal{B}_{next}, S, MIS$);
end if

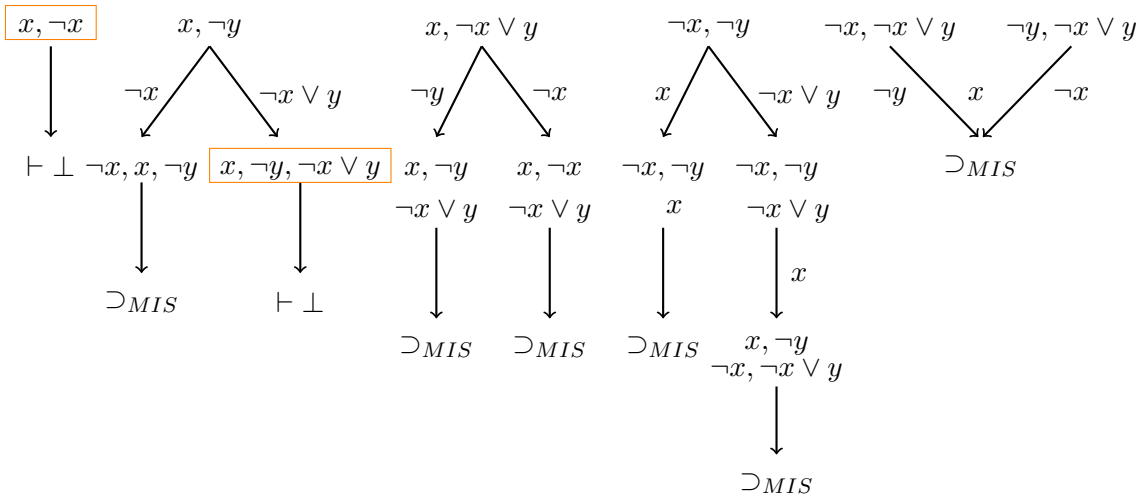

Figure 2: Calcul des sous-bases minimales inconsistantes avec la stratégie Bottom-up

Exemple 3 *Soit* $S = \{x, \neg y, \neg x, \neg x \vee y\}$.
Les sous-bases minimales inconsistantes sont (voir Figure 2):

- $x, \neg x$
- $x, \neg x \vee y, \neg y$

4.3.2 Stratégie Top-down

Cette stratégie présentée dans [13] consiste à commencer la génération par la base entière en essayant d'enlever à chaque itération une formule.

Étant donné une base propositionnelle S, on commence par tester si la base S est inconsistante car si la base S est consistante on arrête la génération. Dans le cas contraire, la première itération consiste à enlever une formule de la base S, puis on recommence sur chaque base tant qu'elle est inconsistante. Si on trouve une base B_i consistante on rajoute $B_i \cup \{\phi_i\}$ (où ϕ_i est la dernière formule enlevée) à l'ensemble des sous-bases minimales inconsistantes MIS, en respectant la minimalité de MIS.

La procédure est résumée dans l'algorithme 7, cet algorithme prend en entrée une base propositionnelle S et donnera en sortie les sous-bases minimales inconsistantes. Cet algorithme fait appel à l'algorithme 8 qui calcule les sous-bases minimales inconsistantes d'une sous-base B_i. Il prend en entrée une sous-base B_i, une formule ϕ, la dernière formule enlevée de la sous-base B_i et l'ensemble des sous-bases minimales inconsistantes MIS. Chaque sous-base minimale inconsistante trouvée sera stockée dans MIS.

Algorithm 7 MI_T(S)

```
1: MIS := ∅;
2: if SAT(S) then
3:     return MIS;
4: else
5:     MIS := Top_down(S, ⊥, ∅);
6: end if
7: return MIS;
```

Algorithm 8 Top_down(B_i, ϕ, MIS))

```
1: if SAT(B_i) then
2:     update(B_i ∪ {φ}, MIS);
3: else
4:     for φ_i ∈ B_i do                          ▷ Choisir une formule φ_i dans B_i
5:         MIS := MIS ∪ Top_down(B_i \ {φ_i}, φ_i, MIS);
6:     end for
7: end if
```

L'algorithme 9 permet de mettre à jour l'ensemble des sous-bases minimales inconsistantes MIS calculées. Il prend en entrée une base propositionnelle B et l'ensemble MIS. En sortie, l'ensemble MIS sera mis à jour.

Algorithm 9 update(B, MIS)

```
1: for B_j ∈ MIS do
2:     if B_j ⊆ B then
3:         return;
4:     else
5:         if B ⊂ B_j then
6:             MIS := MIS \ B_j;
7:         end if
8:     end if
9: end for
10: add B to MIS;
```

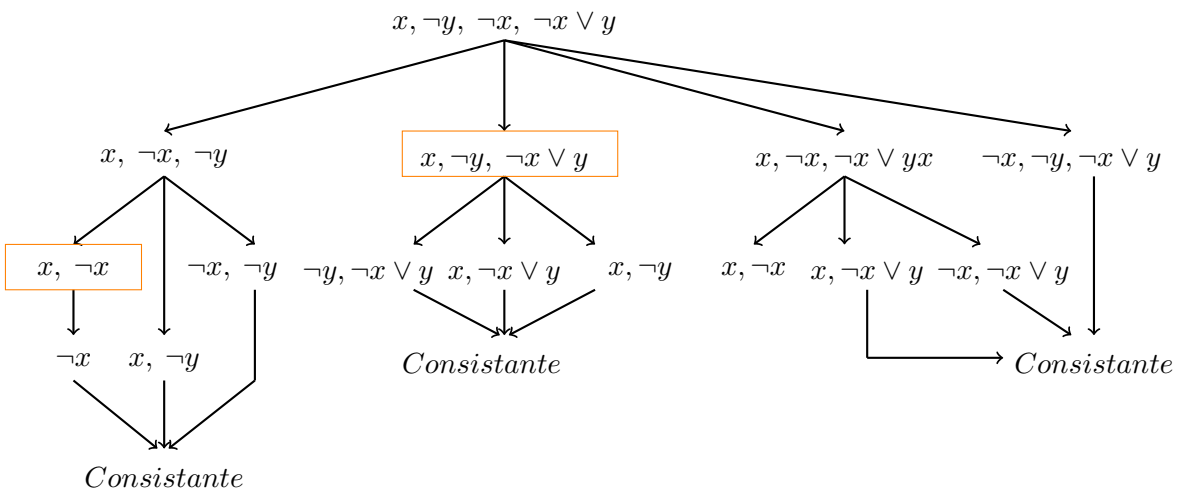

Figure 3: Calcul des sous-bases minimales inconsistantes avec la stratégie Top-down

Exemple 3 (suite) Soit $S = \{x, \neg y, \neg x, \neg x \vee y\}$.
Les bases minimales inconsistantes sont (voir Figure 3) :

- $x, \neg x$
- $x, \neg x \vee y, \neg y$

4.3.3 Calcul à partir des sous-bases maximales consistantes

Une autre stratégie consiste à calculer les sous-bases minimales inconsistantes à partir des sous-bases maximales consistantes (MCS) grâce à la dualité qui existe entre les deux ensembles [14, 15, 16, 17]. L'ensemble des sous-bases maximales consistantes est défini comme suit :

Définition 3 *Soit S une base propositionnelle. $B \subseteq S$ est une sous-base maximale consistante de S si et seulement si*

- $B \nvdash \bot$
- $\forall B' \supset B, B' \vdash \bot$

On note par $MCS(S)$ l'ensemble des sous-bases maximales consistantes de S.

Les auteurs dans [15] ont justifié ce choix de stratégie de calcul par le fait que vérifier la consistance (satisfiabilité) d'une base est plus facile que vérifier son inconsistance. Pour définir les $MIS(S)$ à partir de $MCS(S)$, on utilise la notion de hitting-set introduite par Reiter [18].

Définition 4 *Soit $\mathcal{S}$ une collection d'ensembles. Un hitting-set de $\mathcal{S}$ est un ensemble $H \subseteq \cup_{B \in \mathcal{S}} B$ tel que $H \cap B \neq \emptyset$ pour chaque $B \in \mathcal{S}$. Un hitting-set H de $\mathcal{S}$ est minimal si et seulement si aucun sous-ensemble strict de H n'est un hitting-set de $\mathcal{S}$.*

Définition 5 *Soit S une base propositionnelle. On définit le négatif de l'ensemble $MCS(S)$ par :*

$$MCS^c(S) = \{S \setminus B | B \in MCS(S)\}$$

Proposition 4 *[15] Soit S une base propositionnelle. Les sous-bases minimales inconsistantes de S correspondent aux hitting-sets minimaux de $MCS^c(S)$:*

$$H \in MIS(S) \text{ ssi } H \text{ est un hitting-set minimal de } MCS^c(S)$$

La première étape consiste donc à calculer l'ensemble $MCS(S)$. On peut trouver dans la littérature de nombreux algorithmes [15, 19, 20, 14, 21]. Un algorithme nommé AMC1 [20] nous semble plus efficace, puisque il minimise le nombre d'appels à un solveur SAT. Cet algorithme est basé sur la procédure Davis-Putnam-Logemann-Loveland (DPLL) pour décider la satisfiabilité d'une formule propositionnelle sous forme conjonctive. La procédure est résumée dans l'algorithme 10.
Elle utilise la notion de littéral pur défini comme suit :

Définition 6 *Soit Cl un ensemble de clauses, L est un littéral pur dans Cl si et seulement si $\neg L$ n'est dans aucune clause de Cl.*

Pour éviter de considérer une variable propositionnelle plusieurs fois dans les sous-bases maximales consistantes, on utilise une méthode de marquage. Au départ, toutes les variables sont dans l'état "non-marqué", puis à chaque utilisation d'une variable propositionnelle, son état sera changé en "marqué" pour dire que cette variable a été considérée dans une sous-base maximale, et on refait le même traitement avec une autre variable non-marquée. La procédure est donc décrite comme suit :

1. Un littéral pur n'est jamais une source d'inconsistance. Donc une clause qui contient un littéral pur est présente dans toutes les sous-bases maximales consistantes. La première étape consiste à chercher Δ la sous-base qui contient tous les clauses qui contiennent au moins un littéral pur.

2. Les clauses qui restent contiennent des littéraux non purs, donc pour chaque variable v non-marquée, on construit deux sous-bases maximales, la première contient les clauses qui incluent le littéral v et les clauses de la base Δ, on note cette base Δ_1. De même, on construit une autre sous-base qui contient les clauses qui incluent $\neg v$ et les clauses de la base Δ, on note cette base Δ_2. On marque chaque variable considérée et on refait le même traitement avec une autre variable non-marquée sur les bases Δ_1 et Δ_2 séparément.

3. Si on a exploré toutes les variables de la base (toutes les variables sont marquées), on vérifie que les sous-bases construites sont maximales consistantes (i.e. si on rajoute une formule à une telle base elle devient inconsistante).

On utilise un tableau de booléens ($Marq$) pour marquer les variables. On note par v_ℓ la variable propositionnelle du littéral ℓ. L'algorithme 10 prend en entrée une base propositionnelle sous forme clausale et il donne en sortie l'ensemble des sous-bases maximales consistantes $MCS(S)$. L'algorithme 11 prend en entrée une base propositionnelle sous forme clausale S, une sous-base Δ (une sous-base maximale consistante en construction) et un tableau de marquage de variables $Marq$, et donne en sortie les sous-bases maximales consistantes de S.

Algorithm 10 MCS(S)

$MCS := \emptyset$;
$\mathcal{V}$:= l'ensemble des variables;
$\forall v \in \mathcal{V},\ Marq[v] := false$; ▷ On étiquette toutes les variables de S comme non-marquées
return Find-MC-Subsets(S,$\emptyset$,$Marq$,$\emptyset$);

Algorithm 11 Find-MC-Subsets(S,Δ,$Marq$,MCS)

if $\forall v \in \mathcal{V},\ Marq[v] = true$ **then** ▷ le point 3
  $D := \{\bigvee C | C \in (S \setminus \Delta)\}$;
  if $not(SAT(\Delta \cup D))$ **then**
    add Δ to MCS;
    return MCS;
  end if
else
  if $\exists \ell$ littéral pur $\in S \setminus \Delta$ et $Marq[v_\ell] = false$ **then** ▷ le point 1
    $\Delta := \Delta \cup \{C | C \in (S \setminus \Delta) \wedge \ell$ et $\ell \in C\}$;
    $Marq[v_\ell] := true$;
    return Find-MC-Subsets($S, \Delta, Marq$);
  else ▷ le point 2
    $\exists x$ variable non-marquée
    $Marq(x) := true$;
    $\Delta_1 := \Delta \cup \{C | C \in (S \setminus \Delta)$ et $x \in C\}$;
    $\Delta_2 := \Delta \cup \{C | C \in (S \setminus \Delta)$ et $\neg x \in C\}$;
    MC_1 :=Find-MC-Subsets(S, Δ_1, Mar, MCS);
    MC_2 :=Find-MC-Subsets($S, \Delta_2, Marq, MCS$);
    return $MC_1 \cup MC_2$;
  end if
end if

Exemple 4 *Soit* $\mathcal{K} = \{\neg x \vee z, \neg y \vee z, x, y, z, \neg z\}$ *une base propositionnelle de clauses et* $V = \{x, y, z\}$ *l'ensemble des variables.*
On calcule $MCS(\mathcal{K})$*: Soit* $Marq = [false, false, false]$.
On calcule Find-MC-Subsets($\{\mathcal{K}, \emptyset, [false, false, false]\}$*);*
Il n'y a aucun littéral pur. $\Delta = \emptyset$.
Ligne 12: soit z *variable non-marquée.*
$\Delta_1 = \{\neg x \vee z, \neg y \vee z, z\}$ *et* $\Delta_2 = \{\neg z\}$*;*
On calcule Find-MC-Subsets($\mathcal{K}, \Delta_1, [false, false, true]$*):*
Ligne 7: $\ell = x$ *dans* $\mathcal{K} \setminus \Delta_1 = \{x, y, \neg z\}$ *donc* $Marq = [true, false, true]$.
$\Delta_1 = \{\neg x \vee z, \neg y \vee z, z, x\}$
On calcule Find-MC-Subsets($\mathcal{K}, \Delta_1, [true, false, true]$*):*
Ligne 7: $\ell = y$ *dans* $\mathcal{K} \setminus \Delta_1 = \{y, \neg z\}$ *donc* $Marq = [true, true, true]$.
$\Delta_1 = \{\neg x \vee z, \neg y \vee z, z, x, y\}$
On calcule Find-MC-Subsets($\mathcal{K}, \Delta_1, [true, true, true]$*):*
Ligne 1: $D = \{\neg z\}$, $not(SAT(\Delta_1 \cup D))$. *Donc* $\Delta_1 \in$ *MC.*
On refait la même procédure avec Δ_2*. Donc les sous-bases maximales consistantes de* $\mathcal{K}$ *sont:*

- $MC_1 = \{\neg x \vee z, \neg y \vee z, z, x, y\}$
- $MC_2 = \{\neg z, x, \neg y \vee z\}$
- $MC_3 = \{\neg z, x, y\}$
- $MC_4 = \{\neg z, \neg x \vee z, y\}$
- $MC_5 = \{\neg z, \neg x \vee z, \neg y \vee z\}$

4.3.4 Calcul des hitting-sets minimaux (HSM)

Après avoir calculé l'ensemble des sous-bases maximales consistantes, calculer les sous-bases minimales inconsistantes revient à calculer des hitting-sets [15, 22].

La calcul des hitting-sets minimaux a été largement étudié dans la littérature [18, 23, 22]. Pour notre implémentation nous considérons une heuristique proposée dans [23]. La procédure est inspirée de l'algorithme STACCATO. L'idée est d'attribuer à chaque élément le nombre d'occurrences de cet élément dans la collection d'ensembles dont on calcule les hitting-sets.

Il est facile de voir qu'un élément présent dans tous les ensembles est un hitting-set minimal à lui tout seul. La procédure se base sur ce résultat en commençant par l'élément qui a le plus grand nombre d'occurrences.

Étant donné un ensemble d'ensembles $\mathcal{S}$, on choisit un ensemble X qui a le plus grand score, le score étant défini par la somme des nombres d'occurrences de ses éléments.

- Le calcul des hitting-sets minimaux se fait en choisissant à chaque itération l'élément qui a le plus grand nombre d'occurrences dans X

- Lorsqu'on a traité un élément de X, on le supprime de tous les ensembles de $\mathcal{S}$ qui le contiennent et on recommence sur la collection réduite jusqu'à ce qu'on épuise les éléments de X.

On utilise une liste Occ qui contient les nombres d'occurrences des éléments des ensembles de $\mathcal{S}$, cette liste est mise à jour à chaque itération.

L'algorithme 12 prend en entrée une collection d'ensembles $\mathcal{S}$ et donne en sortie la liste des hitting-sets minimaux. L'algorithme 13 prend en entrée l'ensemble d'ensembles $\mathcal{S}$, le hitting-set en construction HSC, l'élément ayant le plus grand nombre d'occurrences e, les nombres d'occurrences des éléments Occ et l'ensemble des hitting-sets minimaux MIS.

Algorithm 12 ALL_HSM($\mathcal{S}$)

$HIS := \emptyset$;
$Occ :=$ la liste des occurrences des éléments des ensembles de $\mathcal{S}$;
X:= l'élément de $\mathcal{S}$ qui a le plus grand score;
for $e \in X$ avec e l'élément ayant le plus grand nombre d'occurrences **do**
    HSM($\mathcal{S}, \emptyset, e, Occ, HIS$);
    $\mathcal{S} := \{B_i \setminus \{e\} | B_i \in \mathcal{S}\}$;
end for
return HIS;

Algorithm 13 HSM($\mathcal{S}, HSC, e, Occ, HIS$)

if $\mathcal{S} = \emptyset$ **then**
return $\emptyset$
end if
if $Occ(e) = |S|$ **then**
    HIS :=HIS $\cup$HSC $\cup \{e\}\}$;
    $\mathcal{S} := \{B_i \setminus \{e\} | B_i \in \mathcal{S}\}$;
end if
$\mathcal{S}' := \mathcal{S} \setminus \{B_i | e \in B_i\}$;
$Occ' :=$ la liste des occurrences des éléments dans $\mathcal{S}'$;
e' élément l'élément ayant le plus grand nombre d'occurrences dans $\mathcal{S}'$;
return HSM($\mathcal{S}', HSC \cup \{e\}, e', Occ', HIS$);

Exemple 4 (suite) Si on considère l'ensemble MCS de l'exemple 4, on calcule les MIS en calculant les hitting-sets minimaux sur MCS$^c(S)$. On a:

- $\mathcal{K} = \{z, \neg x \vee z, \neg y \vee z, x, y, \neg z\}$
- $MCS(\mathcal{K}) = \{\{z, \neg x \vee z, \neg y \vee z, x, y\}, \{\neg z, x, y\}, \{\neg z, x, \neg y \vee z\}, \{\neg z, \neg x \vee z, y\}, \{\neg z, \neg x \vee z, \neg y \vee z\}\}$

- $\mathcal{S} = MCS^c(\mathcal{K}) = \{\{\neg z\}, \{\neg x \vee z, y, z\}, \{\neg x \vee z, \neg y \vee z, z\}, \{x, \neg y \vee z, z\}, \{x, y, z\}\}$

On calcule ALL_HSM($\mathcal{S}$):
Les nombres d'occurrences des clauses de la base $\mathcal{K}$ dans $\mathcal{S}sontOcc = \{(z, 4), (\neg x \vee z, 2), (\neg y \vee z, 2), (x, 2), (y, 2), (\neg z, 1)\}$.
$X = \{\neg x \vee z, y, z\}$ a le plus grand score : $Occ(\neg x \vee z) + Occ(y) + Occ(z) = 2 + 2 + 4 = 8$).
Soit $e = z$.
On commence par l'élément $(z, 4)$, on calcule HSM($\mathcal{S}, \emptyset, z, Occ, \emptyset$):
$Occ(z)$ est inférieur à la taille de l'ensemble $\mathcal{S}$. Donc on a: $\mathcal{S}' = \{\{\neg z\}\}$ et $Occ' = \{(\neg z, 1)\}$.
On calcule HSM($\mathcal{S}', \{z\}, \neg z, Occ', \emptyset$):
On a $Occ(\neg z) = |S'|$, alors $\{z, \neg z\} \in$ MIS et $\mathcal{S}' = \emptyset$.
On supprime z alors $\mathcal{S} = \{\{\neg z\}, \{\neg x \vee z, \neg y \vee z\}, \{\neg x \vee z, y\}, \{\neg y \vee z, x\}, \{x, y\}\}$
Le prochain élément est $(\neg x \vee z, 2)$ et on refait la même procédure avec l'ensemble réduit. Au final, on trouve ces trois sous-bases minimales inconsistantes:

- $z, \neg z$
- $\neg x \vee z, x, \neg z$
- $\neg y \vee z, y, \neg z$

Parmi les trois stratégies de calcul des sous-bases minimales consistantes, la troisième est la plus efficace puisque la recherche des sous-bases maximales consistantes est un problème plus simple que celle des sous-bases minimales inconsistantes [15]. D'un point de vue des appels au solveur SAT, cette troisième stratégie utilise moins d'appels que les deux autres. C'est donc celle que nous utiliserons pour notre implémentation.

4.4 Inférence d'un couple $\phi > \psi$ en logique possibiliste symbolique

La différence entre l'inférence possibiliste symbolique et l'inférence du système $\mathcal{S}_1$ est qu'en logique possibiliste symbolique (ou standard), on infère une formule et son poids associé. Cependant, l'inférence d'un couple de formules de la forme $\phi > \psi$ est possible dès lors que les deux poids associés aux deux formules sont comparables. Nous rappelons que le degré de nécessité est calculé en utilisant l'équation (6).

Le but de LPosS est de comparer les degrés de certitude d'un couple de formules via les poids associés aux deux formules (les degrés de nécessité dans la fermeture déductive).

Définition 7 *$(\Sigma, \mathcal{C})$ implique que ϕ est plus certain que ψ (notée par $\phi > \psi$) si et seulement si $\mathcal{C} \vDash N_\Sigma(\phi) > N_\Sigma(\psi)$.*

Exemple 5 *Soit $\Sigma = \{(x, p), (\neg x \vee y, q), (\neg x, r), (\neg y, s)\}$ et $\mathcal{C} = \{p > q, q > r, q > s\}$. Alors, $N_\Sigma(y) = \max(\min(p, q), \min(p, r)) = \max(q, r) = q$ et $N_\Sigma(x) = p$. Donc, $(\Sigma, \mathcal{C})$ permet de déduire $x > y$.*

Notons qu'en logique possibiliste, pour comparer deux formules possibilistes en comparant leurs degrés de certitude, par la définition 7 il faut que l'ensemble des contraintes $\mathcal{C}$ soit non vide. Dans le cas où $\mathcal{C}$ est vide, aucune inégalité stricte ne peut être déduite entre les poids des formules.

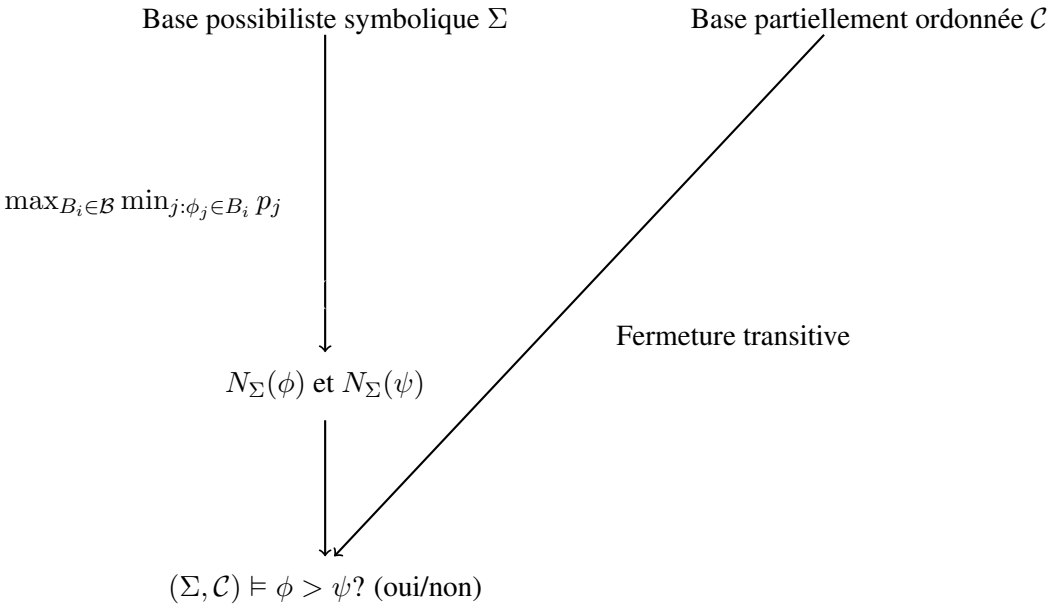

Figure 4: schéma explicatif du processus d'inférence possibiliste symbolique

Étant donné deux formules ϕ, ψ, une première étape consiste à calculer les degrés de nécessité de chaque formule. Donc on obtient deux $\max$ / $\min$ expressions. On cherche à comparer $N_\Sigma(\phi)$ et $N_\Sigma(\psi)$:

$$\max_{B_i \in \mathcal{B}} \min_{j:\phi_j \in B_i} p_j > \max_{B_j \in \mathcal{B}} \min_{i:\phi_i \in B_j} p_i$$

est de la forme:

$$\begin{array}{rcl} \max(\min(a_1, \cdots a_n) & & \max(\min(b_1, \cdots b_m) \\ \quad . & & \quad . \\ \quad . & > & \quad . \\ \quad . & & \quad . \\ \min(a'_1, \cdots a'_p)) & & \min(b'_1, \cdots b'_q)) \end{array}$$

Donc $N_\Sigma(\phi) > N_\Sigma(\psi)$ si et seulement si pour toute expression $\min(b_i, \cdots, b_m)$ dans $N_\Sigma(\psi)$, il existe une expression $\min(a_1, \cdots a_n)$ dans $N_\Sigma(\phi)$ telle que $\min(a_1, \cdots a_n) > \min(b_i, \cdots, b_m)$.

De plus une expression de la forme $\min(E_1)$ domine une autre expression de la même forme $\min(E_2)$ où E_1 et E_2 sont des sous ensembles de poids, si et seulement si $\forall p \in E_1, \exists q \in E_2, p > q$.

L'algorithme 14 décide si une expression $\min$ domine une autre. Cet algorithme prend deux ensembles de poids F et G et un ensemble de contraintes sur les poids symboliques $\mathcal{C}$. Il donne en sortie la décision si F domine G ou non.

Algorithm 14 $Comp\_Min(F, G, \mathcal{C})$

```
1: Dec:=false;
2: while Dec=false and b_i ∈ G do
3:     Dec:=true;
4:     while Dec=true and a_i ∈ F do
5:         Dec:=Dec ∧ a_i > b_i ∈ C;
6:     end while
7: end while
8: return Dec;
```

L'algorithme 15 décide si une expression $\max$ domine une autre. Il prend en entrée $\mathcal{F}$ et $\mathcal{G}$ deux familles d'ensembles de poids et un ensemble de contraintes sur les poids symboliques $\mathcal{C}$. Il donne en sortie la décision si $\mathcal{F}$ domine $\mathcal{G}$ ou non.

Algorithm 15 $Comp\_Max(\mathcal{F}, \mathcal{G})$

```
1: Dec:=false;
2: while Dec=false and E_j ∈ F do
3:     Dec:=true;
4:     while Dec=true and E_i ∈ G do
5:         Dec:=Dec ∧ Comp_Min(E_i, E_j, C);
6:     end while
7: end while
8: return Dec;
```

Exemple 6 *Soit* $\Sigma = \{(\neg x \vee y, a), (x, b), (\neg y, c), (y, d), (\neg x, e)\}$ *une base possibiliste symbolique avec* $\mathcal{C} = \{a > d, d > e, b > e, c > e, d > c\}$.
On cherche à comparer $N_\Sigma(y)$ *et* $N_\Sigma(\neg x)$. *On calcule* $N_\Sigma(y)$*:*
Soit $\Sigma^* = \{\neg x \vee y, x, y, \neg y, \neg x\}$. *Les éléments de MIS(*$\Sigma^* \cup \{\neg y\}$*) qui contiennent* $\neg y$ *sont:*

- $y, \neg y$
- $\neg x \vee y, x, \neg y$

Les éléments de MIS(Σ^**) qui ne contiennent aucune* $B_i \in$ *MIS(*$\Sigma^* \cup \{\neg y\}$*) sont:*

- $x, \neg x$

$$\begin{aligned} N_\Sigma(y) &= \max(d, \min(a, b), \min(b, e))) \\ &= \max(d, \min(a, b)) \end{aligned}$$

On calcule maintenant $N_\Sigma(\neg x)$*:*
Les éléments de MIS($\Sigma^* \cup \{x\}$*) qui contiennent* x *sont:*

- $x, \neg x$
- $\neg x \vee y, x, \neg y$

Les éléments de MIS(Σ^**) qui ne contiennent aucune* $B_i \in$ *MIS(*$\Sigma^* \cup \{x\}$*) sont:*

- $y, \neg y$

$$\begin{aligned} N_\Sigma(\neg x) &= \max(e, \min(a, c), \min(c, d))) \\ &= c \end{aligned}$$

On conclut que $y > \neg x$ *puisque* $\max(d, min(a, b)) \geq d > c$*.*

PlutàŽt que d'appliquer ces tests de manière brute, il est naturel d'utiliser les ensembles de poids élémentaires au lieu de formules, et de simplifier les expressions de $N_\Sigma(\phi)$ et $N_\Sigma(\psi)$ avant de les comparer. La technique présentée dans la section suivante est utile à cet effet.

4.5 Inférence syntaxique basée sur les ATMS

Dans cette section nous présentons une méthode d'inférence pour la logique possibiliste symbolique inspirée d'une approche utilisée pour faire du raisonnement abductif. Cette approche utilise les concepts des ATMS [24, 25].

4.5.1 Rappel sur les ATMS (Assumption-base Truth Maintenance Systems)

Dans cette section, nous rappelons les concepts de base des ATMS. Un ATMS a pour but de gérer les interdépendances dans une base de connaissances. Les mécanismes spécifiques d'un ATMS sont fondés sur la distinction entre deux ensembles de données <$\mathcal{J}, \mathcal{A}$> (représentées par des variables propositionnelles), où $\mathcal{A}$ est un ensemble d'hypothèses et $\mathcal{J}$ est une base de justifications (connaissances). L'idée de base est que les hypothèses sont des données primitives, toutes les autres données peuvent en être dérivées. Ce sont les paramètres qui caractérisent les différentes situations pour un problème donné, ce problème est décrit par un ensemble de clauses appelés justifications. Tout ensemble d'hypothèses peut caractériser une situation.

Définition 8 (environment) *On appelle environnement tout sous-ensemble* $E \subset \mathcal{A}$*.*

La principale vocation d'un ATMS est de répondre à deux questions. La première est la suivante: "Étant donné un ensemble de justifications, quels sont les environnements menant à une contradiction"? On répond à cette question par le concept de $nogood$.

Définition 9 (Nogood) *Un nogood E est un environnement incohérent minimal, c'est à dire : E est minimal pour l'inclusion tel que $\mathcal{J} \cup E \vdash \bot$*

La deuxième question est: "Sur la base de l'ensemble des justifications, quels environnements permettent de croire en une donnée"? On répond à cette question par le concept de label.

Définition 10 (Label) *Soit <$\mathcal{J}, \mathcal{A}$> la base d'un ATMS et ϕ une donnée. $Label(\phi)$ est un ensemble d'environnements tel que:*

- *aucun $E_i \in Label(\phi)$ ne contient un nogood (Cohérence);*
- *si $E_i \in Label(\phi)$ alors $\mathcal{J} \cup E_i \vdash \phi$ (Correction);*
- *$\forall E_i$, tel que $\mathcal{J} \cup E_i \vdash \phi$, E_i contient un élément de $Label(\phi)$ (Complétude);*
- *$\forall E_i \in Label(\phi)$, $\nexists E_j \in Label(\phi)$ et $E_j \subset E_i$ (Minimalité).*

Notons que l'ATMS de base a été étendu pour pouvoir calculer le label d'une formule propositionnelle quelconque, via un codage par des hypothèses supplémentaires.

4.5.2 Codage d'une base possibiliste symbolique et adaptation

L'idée est de considérer les poids impliqués dans le calcul de $N_\Sigma(\phi)$ comme des hypothèses qui expliquent la certitude de ϕ.

Étant donné $(\Sigma, \mathcal{C})$, la base possibiliste Σ est codée par une paire ($\mathcal{J}$, $\mathcal{A}$) comme suit : chaque poids élémentaire a_i est associé à une variable propositionnelle (pour simplifier, nous notons aussi a_i la variable propositionnelle correspondante) et chaque formule possibiliste (ϕ_i, a_i) est codée par la formule propositionnelle $\neg a_i \vee \phi_i$.

Définition 11 *Soit $(\Sigma, \mathcal{C})$ une base possibiliste symbolique. La base ATMS associée est <$\mathcal{J}_\Sigma, \mathcal{A}$> où :*

- $\mathcal{J}_\Sigma = \{\neg p \vee \phi | (\phi, p) \in \Sigma\}$
- $\mathcal{A} = \{p_i | (\psi_i, p_i) \in \Sigma\}$

Dans le but de calculer $N_\Sigma(\phi)$, nous devons calculer les sous-bases de Σ^* qui sont minimales consistantes impliquant ϕ, et puis certaines des sous-bases inconsistantes minimales de Σ^*. De plus, pour calculer $N_\Sigma(\phi)$, nous avons seulement besoin des poids associés aux formules appartenant à ces sous-bases.
Avec le codage de la définition 11, il est facile de voir que chaque sous-base consistante de Σ^* qui

est minimale impliquant ϕ correspond exactement à un environnement dans $Label(\phi)$ par rapport à la base ($\mathcal{J}_\Sigma$, $\mathcal{A}$). Et chaque sous-base inconsistante minimale de Σ^* correspond exactement à un $nogood$ par rapport à la base ($\mathcal{J}_\Sigma$, $\mathcal{A}$).

Proposition 5 *Soit $(\Sigma, \mathcal{C})$ une base possibiliste symbolique et la base ATMS associée ($\mathcal{J}_\Sigma$, $\mathcal{A}$). Soit $\mathcal{U}(\phi) = \{U_1, \cdots, U_k\}$ l'ensemble des* nogoods utiles *pour ϕ, c.à .d. les nogoods qui ne contiennent pas d'environnement de $Label(\phi)$.*

$N_\Sigma(\phi) = \max(\max_{E \in Label(\phi)} \min_{a \in E} a, \max_{i=1}^{k} \min_{a \in U_i} a)$

Exemple 6 (suite) On considère la même base possibiliste symbolique
$\Sigma = \{(\neg x \vee y, a), (x, b), (\neg y, c), (y, d), (\neg x, e)\}$ avec $\mathcal{C} = \{a > d, d > e, b > e, c > e, d > c\}$.
Cette base sera traduite par:

- $\mathcal{J}_\Sigma = \{\neg a \vee \neg x \vee y, \neg b \vee x, \neg c \vee \neg y, \neg d \vee y, \neg e \vee \neg x\}$
- $\mathcal{A} = \{a, b, c, d, e\}$.

Calculons $N_\Sigma(y)$:
Par la proposition 5, $N_\Sigma(y) = \max(\max_{E \in Label(y)} \min_{a \in E} a, \max_{i=1}^{k} \min_{a \in U_i} a)$.
On a :

- $Label(y) = \{\{d\}, \{a, b\}\}$
- $\mathcal{U}(y) = \{\{b, e\}\}$

Donc $N_\Sigma(y) = \max(d, \min(a, b), \min(b, e))$.

Il reste ensuite à comparer les poids symboliques en utilisant la procédure présentée dans la section précédente.

4.6 Comparaison avec l'inférence possibiliste symbolique

L'un des avantages de la dernière méthode réside dans le fait que tout est calculé seulement en termes de poids (dans le label de la formule et les nogoods utiles). Puis, les contraintes sur les poids sont utilisées à la fin pour simplifier les expressions $\max$ / $\min$, alors que dans la méthode de la section 4.4, nous utilisons toutes les formules de la base possibiliste symbolique. De plus, dans la méthode ATMS, on peut penser à exploiter les contraintes pour simplifier les ensembles de poids impliqués dans la comparaison des degrés de nécessité au moment du calcul des ensembles (label et nogoods utiles).

Il est donc naturel de simplifier les expressions $N_\Sigma(\phi)$ et $N_\Sigma(\psi)$ avant de les comparer,

- en remplaçant chaque ensemble de poids $B \in Label(\phi) \cup \mathcal{U}(\phi)$ par l'ensemble réduit de poids $W = \min_{\mathcal{C}}(B)$ constitué par les poids minimaux dans B selon l'ordre partiel défini par les contraintes $\mathcal{C}$,

- ensuite par la suppression des ensembles W dominés, dans le sens ou $\mathcal{C} \models \min\{a \in W'\} > \min\{a \in W\}$ par l'algorithme 14.

Bien sà»r, nous pouvons appliquer ces simplifications dès que les éléments des labels ou nogoods utiles sont produits.

Exemple 7 *Considérons la base possibiliste* $\Sigma = \{(\neg x \vee y, a), (x, b), (\neg y, c), (\neg x, e)\}$ *avec* $\mathcal{C} = \{a > b, a > c, b > e, c > e\}$. *Nous voulons vérifier si* $\mathcal{C} \models N_\Sigma(y) > N_\Sigma(\neg x)$.
Notons que $Label(y) = \{\{a, b\}\}$ *et* $\mathcal{U}(y) = \{\{b, e\}\}$. *De même on calcule* $Label(\neg x) = \{\{a, c\}, \{e\}\}$, *concernant les nogoods, aucun n'est utile.*
En utilisant $\mathcal{C}$ *on peut réduire* $\{a, b\}$ *à* $\{b\}$ *et* $\{b, e\}$ *à* $\{e\}$ *et le degré de nécessité de* y *à* b, *puisque* $b > e \in \mathcal{C}$. *De même, on peut réduire* $\{a, c\}$ *à* $\{c\}$ *et le degré de nécessité de* $\neg x$ *à* c *puisque* $c > e \in \mathcal{C}$. *On a* $b > c \notin \mathcal{C}$, *nous ne pouvons pas conclure que* $y > \neg x$ *(ni l'inverse).*

En général, la suppression des ensembles de poids dominés peut être obtenue au moyen de l'algorithme 14 appliqué à toutes les paires d'ensembles réduits dans $Label(\phi) \cup \mathcal{U}(\phi)$.
Finalement, nous pouvons comparer l'ensemble des sous-ensembles réduits non-dominés de $Label(\phi) \cup \mathcal{U}(\phi)$ et de $Label(\psi) \cup \mathcal{U}(\psi)$, dans le but de décider si $\phi > \psi$, en utilisant l'algorithme 15.

5 Conclusion

Ce rapport est une autre étape dans l'étude de l'inférence d'une base propositionnelle partiellement ordonnée. Nous avons présenté l'implémentation de deux systèmes d'inférence à partir de bases partiellement ordonnées qui appartiennent à deux familles de logiques différentes. Nous avons d'abord présenté une méthode de preuve pour la logique du système $\mathcal{S}_1$ qui est un fragment de logique conditionnelle. Nous avons montré comment utiliser les règles d'inférences de ce système afin de raisonner sur des bases partiellement ordonnées.

Nous avons aussi présenté une version de la logique possibiliste avec des poids symboliques partiellement ordonnés avec son implémentation. Cette logique est différente du cadre des logiques conditionnelles [2, 3] par l'utilisation du principe de spécificité minimale qui n'est pas mis en œuvre dans ces cadres logiques, et qui est plus productif.

Deux méthodes d'inférence syntaxiques pour cette logique sont définies qui nous permettent de déduire de nouvelles formules avec des poids symboliques complexes (degrés de nécessité de formules). La première méthode nécessite l'énumération des sous-ensembles minimaux inconsistants pour calculer les degrés de nécessité. La deuxième méthode se base sur l'utilisation du formalisme ATMS. Elle permet de prendre en compte les contraintes sur les poids d'une manière plus directe et facilite la comparaison des degrés de nécessité symboliques.

Une possible amélioration de la méthode d'inférence basée sur les ATMS consisterait prendre en compte les contraintes sur les poids symboliques plus tôt dans le processus d'inférence. L'idée la plus simple est de coder les contraintes sur les poids symboliques par des formules logiques et de les intégrer dans la base des justifications. Les auteurs dans [5] ont trouvé un codage logique

pour des contraintes larges ($\geq$), ce qui n'est pas le cas dans notre cadre. Il n'est pas clair qu'on puisse le faire pour des contraintes strictes.

Ce travail a des applications potentielles pour la révision et la fusion de croyances, ainsi que la modélisation des préférences [26].

References

[1] Lewis, D.: Counterfactuals and comparative possibility. Journal of Philosophical Logic **2** (1973) 418–446

[2] Halpern, J.Y.: Defining relative likelihood in partially-ordered preferential structures. Journal of Artificial intelligence Research **7** (1997) 1–24

[3] Cayrol, C., Dubois, D., Touazi, F.: On the deductive closure of a partially ordered propositional belief base (Huitièmes journées d'Intelligence Artificielle Fondamentale, Angers. (2014)

[4] Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence **44** (1990) 167–207

[5] Benferhat, S., Prade, H.: Encoding formulas with partially constrained weights in a possibilistic-like many-sorted propositional logic. In Kaelbling, L.P., Saffiotti, A., eds.: IJCAI, Professional Book Center (2005) 1281–1286

[6] Cayrol, C., Dubois, D., Touazi, F.: On the semantics of partially ordered bases. In: Foundations of Information and Knowledge Systems - 8th International Symposium. Lecture Notes in Computer Sciences, Springer (2014) 136–153

[7] Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence **44** (1990) 167 – 207

[8] Harman, G.: Practical aspects of theoretical reasoning. The Oxford Handbook of Rationality (2004) 45–56

[9] Dubois, D., Prade, H.: Possibilistic logic: a retrospective and prospective view. Fuzzy Sets and Systems **144** (2004) 3–23

[10] Cayrol, C., Dubois, D., Touazi, F.: Fermeture déductive d'une base partiellement ordonnée. Rapport de recherche RR–2014-08–FR, IRIT, Université Paul Sabatier, Toulouse (2014)

[11] Grégoire, r., Mazure, B., Piette, C.: Sous-formules minimales insatisfaisables. Hermes (2008)

[12] Lange, M., Leiß, H.: To cnf or not to cnf? an efficient yet presentable version of the cyk algorithm. Informatica Didactica **8** (2009) 2008–2010

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

[14] Grégoire, E., Mazure, B., Piette, C.: Using local search to find {MSSes} and {MUSes}. European Journal of Operational Research **199** (2009) 640 – 646

[15] Kevin McAreavey, W.L., Miller, P.: Computational approaches to finding and measuring inconsistency in arbitrary knowledge bases. International Journal of Approximate Reasoning **55** (2014) 1659 – 1693

[16] Liffiton, M., Sakallah, K.: On finding all minimally unsatisfiable subformulas. In: Theory and Applications of Satisfiability Testing. Volume 3569 of Lecture Notes in Computer Science. (2005) 173–186

[17] Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Practical Aspects of Declarative Languages. Springer (2005) 174–186

[18] Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32** (1987) 57–95

[19] de la Banda, M.G., Stuckey, P.J., Wazny, J.: Finding all minimal unsatisfiable subsets. In: Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming, ACM (2003) 32–43

[20] Birnbaum, E., Lozinskii, E.L.: Consistent subsets of inconsistent systems: structure and behaviour. Journal of Experimental and Theoretical Artificial Intelligence **15** (2003) 25–46

[21] Grégoire, E., Mazure, B., Piette, C.: Extracting muses. In: ECAI'06: 17th European Conference on Artificial Intelligence, Riva Del Garda. (2006) 387–391

[22] Liffiton, M., Sakallah, K.: Algorithms for computing minimal unsatisfiable subsets of constraints. Journal of Automated Reasoning **40** (2008) 1–33

[23] Abreu, R., van Gemund, A.J.C.: A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. (2009) 2–9

[24] De Kleer, J.: A general labeling algorithm for assumption-based truth maintenance. In: American Association for Artificial Intelligence. (1988) 188–192

[25] De Kleer, J.: An assumption-based tms. Artificial intelligence **28** (1986) 127–162

[26] Dubois, D., Prade, H., Touazi, F.: Conditional preference nets and possibilistic logic. In van der Gaag, L.C., ed.: Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Volume 7958 of Lecture Notes in Computer Science., Springer (2013) 181–193