Calcul de conséquences dans un système d'inférence pair-à-pair propositionnel

Consequence Finding in a Propositional Peer-to-peer Inference System

Nada Abdallah François Goasdoué

Univ. Paris-Sud & CNRS (LRI/IASI) – INRIA (Saclay/GEMO)

LRI, bâtiment 490, Université Paris-Sud, 91405 Orsay Cedex
{nada,fg}@lri.fr

Résumé

Dans cet article, nous étudions le calcul de conséquences dans les systèmes d'inférence pair-à-pair (P2PIS) propositionnels avec des mappings orientés. Dans ces systèmes, un mapping allant d'un pair vers un autre spécifie un ensemble de connaissances que le premier pair doit observer, ainsi que les connaissances qu'il doit notifier au second pair si les connaissances observées sont satisfaites. Ces nouveaux P2PIS pouvant modéliser de nombreuses applications réelles, il est important de les doter d'inférences clés de l'IA, en l'occurrence du calcul de conséquences.

Nos contributions sont doubles. Nous définissons tout d'abord le premier cadre logique pour représenter des P2PIS propositionnels avec des mappings orientés. Nous étudions ensuite le calcul de conséquences dans ce nouveau cadre. En particulier, nous proposons un algorithme totalement décentralisé pour ce problème.

Mots Clefs

IA distribuée, déduction automatique, logique propositionnelle.

Abstract

In this paper, we study consequence finding in propositional peer-to-peer inference systems (P2PISs) with directed mappings. In those systems, a mapping from a peer to another specifies some knowledge that the former peer has to observe and the knowledge it must notify to the latter peer if the observed knowledge is satisfied. Such new P2PISs can model many real distributed applications and therefore studying the basic AI task of consequence finding in such systems is of great interest.

Our contributions are to define the first logical setting for propositional P2PISs with directed mappings, and to study consequence finding in that new setting. In particular, we design a fully decentralized algorithm for that problem.

Keywords

Distributed AI, automated deduction, propositional logic.

1 Introduction

Le *calcul de conséquences* consiste à dériver certains théorèmes implicites à une base de connaissances. Ces théorèmes sont, par exemple, ceux en terme d'un langage donné, ou encore ceux résultant de l'ajout de nouvelles informations dans la base de connaissances. Ce calcul joue un rôle important en IA car de nombreuses inférences de ce domaine peuvent s'y ramener : le raisonnement de sens commun, le diagnostic à base de modèles, la compilation de bases de connaissances, etc.

Le lecteur intéressé par les fondements du calcul de conséquences pourra se référer à [23] qui offre un panorama des principaux résultats et applications en logique propositionnelle, ainsi qu'à [24, 20, 21] pour ce qui concerne la logique du premier ordre, à [7] pour les logiques de description et enfin à [8] pour les logiques modales.

Ces dernières années, le calcul de conséquences a été principalement étudié en logique propositionelle et sous l'angle de l'optimisation. En particulier, [4, 5, 6] proposent de commencer par partitionner une théorie de la logique propositionnelle (ou de la logique du premier ordre) dans laquelle il faut effectuer un calcul de conséquences, puis d'exploiter astucieusement son partionnement afin d'effectuer un calcul très efficace. Une autre approche remarquable, développée dans [28], utilise les ZBDDs (technique de compression pour de grands ensembles de clauses) afin de faire passer le calcul de conséquences à l'échelle de théories logiques jusqu'alors intraitables du fait du nombre trop important de leurs théorèmes.

Récemment, les systèmes d'inférence pour le calcul de conséquences ont évolué vers les *systèmes d'inférence pair-à-pair* dans [1, 2, 3].

Les systèmes d'inférence pair-à-pair (P2PIS) sont constitués de pairs autonomes (c.-à-d. conçus et gérés indépendamment les uns des autres) qui peuvent communiquer afin de réaliser une inférence particulière au niveau du système global. Dans ces systèmes, les règles de communication entre pairs sont modélisées par des *mappings* définissant des relations sémantiques entre les connaissances des pairs.

Un aspect crucial de ce nouveau cadre distribué vient du fait qu'aucun pair, et donc aucun acteur du système, n'a une vision globale d'un P2PIS. Chaque pair ne connait que les connaissances qu'il gère et ses mappings avec d'autres pairs. Ceci soulève des problèmes algorithmiques intéressants et non triviaux puisque les algorithmes de raisonnement de la littérature ont été développés avec l'hypothèse que toutes les connaissances – sur lesquelles les inférences doivent être réalisées – sont fournies en entrée. De nouveaux algorithmes décentralisés doivent donc être définis avec l'idée que seul un sous-ensemble des connaissances globales est disponible en entrée d'un pair (c.-à-d. ses connaissances et ses mappings), mais que les inférences réalisées doivent tout de même être correctes et complètes par rapport aux connaissances globales du système (c.-à-d. les connaissances et les mappings de tous les pairs).

Dans les P2PIS proposés par [1, 2, 3], chaque pair gère une théorie clausale propositionnelle en termes de ses propres variables propositionnelles. Un pair établit (ou supprime) un mapping en ajoutant à (ou en supprimant de) sa théorie une clause en termes de ses variables et de celles d'autres pairs, ces derniers étant informés de l'opération effectuée.
Un point central de la définition de ces P2PIS est que les mappings sont *non-orientés*, c.-à-d. que tout pair impliqué dans un mapping peut utiliser ce dernier pour propager des connaissances aux autres pairs participant au mapping.
Dans cette approche, le calcul de conséquences est fondé sur la résolution (*cf.* [25, 11]) dont de nombreuses stratégies ont été étudiées pour ce calcul dans un cadre centralisé (*cf.* [24, 20, 21, 23]). Plus précisément, [1, 2, 3] proposent l'algorithme décentralisé DECA – effectuant une procédure décentralisée de résolution – qui calcule *tous* les *impliqués clausaux premiers propres* (c.-à-d. les conséquences les plus fortes) d'une clause fournie en entrée d'un pair, par rapport aux connaissances globales d'un P2PIS.

Dans cet article, nous revisitons le problème de calcul de conséquences de [1, 2, 3] dans des P2PIS propositionnels avec des mappings *orientés*. Dans ces systèmes, un mapping est établi entre deux pairs, mais seul l'un des pairs peut utiliser ce mapping pour propager des connaissances à l'autre pair. Plus précisément, un mapping allant d'un pair vers un autre spécifie un ensemble de connaissances que le premier pair doit observer, ainsi que les connaissances qu'il doit notifier au second pair si les connaissances observées sont satisfaites. Ces nouveaux P2PIS sont intéressants pour l'utilisation d'inférences classiquement étudiées en IA car ils peuvent modéliser de nombreuses applications réelles comme, par exemple, des fonctions distribuées dans les véhicules en Ingénierie Automobile ou encore des systèmes de contrôles distribués de processus ou machines industrielles en Automatique.

On pourrait penser que calculer des conséquences dans un P2PIS avec des mappings orientés ou non est assez semblable, la première intuition étant que dans les deux cas les conséquences d'une clause fournie en entrée d'un pair sont calculées par rapport aux connaissances de ce pair et des pairs qui lui sont (transitivement) accessibles via les mappings. Mais en fait, calculer des conséquences dans un P2PIS avec des mappings orientés se révèle plus complexe car tout pair impliqué dans un calcul de conséquences doit aussi prendre en compte les connaissances additionnelles dues aux éventuelles notifications fournies par les mappings. Par conséquent, une difficulté supplémentaire pour le calcul de conséquences dans un P2PIS avec des mappings orientés est de savoir décider si un mapping fournit de telles connaissances additionnelles, c.-à-d. si les connaissances observées définies par le mapping sont satisfaites.

Nos contributions par rapport au calcul de conséquences sont doubles. Tout d'abord, nous définissons le premier cadre logique pour des P2PIS propositionnels avec des mappings oritentés. Nous revisitions ensuite, dans ce cadre, le problème de calcul de conséquences de [1, 2, 3] d'un point de vue théorique, puis algorithmique.
Nous commençons par présenter la logique propositionnelle épistémique que nous utilisons ensuite pour définir les P2PIS. En particulier, l'opérateur épistémique $\mathbf{K}$ nous permet de modéliser des mappings orientés. L'idée d'utiliser cette logique est inspirée de [12, 16], où la logique épistémique du premier ordre est utilisée afin de définir des flux de données allant d'un pair vers un autre dans le cadre de l'interrogation de systèmes pair-à-pair de gestion de données relationnelles.
Nous poursuivons par l'étude du calcul de conséquences dans ces nouveaux P2PIS, pour lequel nous fournissons l'algorithme totalement décentralisé $\text{DECA}_{\mathbf{K}}$.
Enfin, nous concluons par une brève présentation des travaux connexes de la littérature et donnons les perspectives de ce travail.

2 La logique épistemique propositionnelle

La logique épistemique propositionnelle étend la logique propositionnelle avec l'opérateur épistemique $\mathbf{K}$.

Syntaxe Étant donnés un alphabet $\mathcal{A}$ de variables propositionnelles et les constantes $true$ et $false$, les *formules* de la logique épistémique propositionnelle en terme de $\mathcal{A}$ sont définies par :
– $true$ ou $false$ sont des formules,
– $V \in \mathcal{A}$ est une formule,
– $\neg f$, où f est une formule, est une formule,
– $(f_1 \vee f_2)$, où f_1 et f_2 sont des formules, est une formule,
– $(f_1 \wedge f_2)$, où f_1 et f_2 sont des formules, est une formule
– et $\mathbf{K}(f)$, où f est une formule, est une formule.

De plus, $(f_1 \Rightarrow f_2)$ et $(f_1 \Leftrightarrow f_2)$, où f_1 et f_2 sont des formules, sont des formules définies respectivement par $(\neg f_1 \vee f_2)$ et $((f_1 \Rightarrow f_2) \wedge (f_2 \Rightarrow f_1))$.
Enfin, une *théorie* de la logique épistémique propositionnelle en terme d'un alphabet donné est un ensemble d'*axiomes* qui sont des formules de la logique épistémique propositionnelle en terme de cet alphabet.

Sémantique La sémantique de la logique épistémique propositionnelle est fondée sur la notion d'*interprétation épistémique*. Une interprétation épistémique est une paire $(I, \mathcal{W})$, où $\mathcal{W}$ est un ensemble d'*interprétations propositionnelles* et $I \in \mathcal{W}$. Une interprétation propositionnelle I assigne chaque variable V d'un alphabet $\mathcal{A}$ à $true$ ou $false$. La notion de *satisfaction d'une formule* f par une interprétation épistémique $(I, \mathcal{W})$, notée $(I, \mathcal{W}) \models f$, est définie par :
– $(I, \mathcal{W}) \models true$, $(I, \mathcal{W}) \not\models false$,
– $(I, \mathcal{W}) \models V \in \mathcal{A}$ ssi $I(V) = true$,
– $(I, \mathcal{W}) \models \neg f$ ssi $(I, \mathcal{W}) \not\models f$,
– $(I, \mathcal{W}) \models (f_1 \vee f_2)$ ssi $(I, \mathcal{W}) \models f_1$ ou $(I, \mathcal{W}) \models f_2$,
– $(I, \mathcal{W}) \models (f_1 \wedge f_2)$ ssi $(I, \mathcal{W}) \models f_1$ et $(I, \mathcal{W}) \models f_2$
– et $(I, \mathcal{W}) \models \mathbf{K}(f)$ ssi pour tout $J \in \mathcal{W} : (J, \mathcal{W}) \models f$.
Un *modèle épistémique d'une formule* f est une interprétation épistémique qui satisfait f. Un *modèle épistémique d'une théorie* est une interprétation épistémique qui satisfait chaque axiome de cette théorie, et un axiome est satisfait dans $(I, \mathcal{W})$ s'il est satisfait dans tout $(J, \mathcal{W})$ tel que $J \in \mathcal{W}$.
Notons que cette sémantique est celle d'un système modal propositionnel $S5$ (*cf.* [14, 19]), dont la relation d'accessibilité est réflexive, transitive et euclidienne (c.-à-d. une relation d'équivalence).

Pour finir, nous rappelons la notion de (relation de) *conséquence* dans la logique épistémique propositionnelle.

Définition 1 (Relation de conséquence) *Étant donnés un alphabet $\mathcal{A}$ de variables propositionnelles, une théorie $\mathcal{T}$ de la logique épistémique propositionnelle constituée de formules en terme de $\mathcal{A}$, et deux formules f et g de la logique épistémique propositionnelle en terme de $\mathcal{A}$:*
– *f est une* conséquence *de g ($g \models f$) ssi tout modèle épistémique de g est un modèle épistémique de f.*
– *f est une* conséquence *de $\mathcal{T}$ ($\mathcal{T} \models f$) ssi tout modèle épistémique de $\mathcal{T}$ est un modèle épistémique de f.*

3 P2PIS avec mappings orientés

Nous définissons maintenant les P2PIS avec mappings orientés basés sur la logique épistémique propositionnelle.

Syntaxe Un P2PIS $\mathcal{S}$ est un ensemble de pairs.

Un *pair*, noté $\mathcal{P}_i$, gère des connaissances modélisées par une théorie de la logique propositionnelle, notée $\mathcal{T}(\mathcal{P}_i)$, et des règles de communication avec d'autres pairs qui sont définies par un ensemble de *mappings*, noté $\mathcal{M}(\mathcal{P}_i)$.

$\mathcal{T}(\mathcal{P}_i)$ est un ensemble de *clauses* – autrement dit, une forme normale conjonctive – en terme de l'*alphabet* de $\mathcal{P}_i$, noté $\mathcal{A}(\mathcal{P}_i)$, ce dernier étant constitué de *variables* propositionnelles. Nous distinguons un sous-ensemble $cible(\mathcal{A}(\mathcal{P}_i))$ de $\mathcal{A}(\mathcal{P}_i)$ représentant les *variables cibles* du pair, c.-à-d. les variables d'intérêt pour une application donnée (par exemple les variables de bon ou mauvais fonctionnement dans une application de diagnostic à base de modèles). Notons que les alphabets des pairs sont *distincts*. Nous supposons ici que chaque pair a un identifiant unique (par exemple son adresse IP) et que ses variables utilisent cet identifiant. Pour simplifier, nous utilisons comme identifiant d'un pair $\mathcal{P}_i$ l'indice i et nous notons une variable A de $\mathcal{P}_i$ par A_i. Le *langage* de $\mathcal{P}_i$, noté $\mathcal{L}(\mathcal{P}_i)$, est l'ensemble fini de clauses sans répétition de littéral en terme de $\mathcal{A}(\mathcal{P}_i)$. Le *langage cible* de $\mathcal{P}_i$, noté $cible(\mathcal{L}(\mathcal{P}_i))$, est l'ensemble fini de clauses sans répétition de littéral en terme de $cible(\mathcal{A}(\mathcal{P}_i))$. Nous supposons que la clause vide $\square$ appartient à $\mathcal{L}(\mathcal{P}_i)$ et $cible(\mathcal{L}(\mathcal{P}_i))$.
Les mappings dans lesquels $\mathcal{P}_i$ est impliqué sont stockés localement dans $\mathcal{M}(\mathcal{P}_i)$. Il en découle que tout mapping entre deux pairs est stocké dans ces deux pairs. Chaque mapping de $\mathcal{P}_i$ est de la forme $\mathbf{K}(l_i) \Rightarrow l_j$ ou $\mathbf{K}(l_j) \Rightarrow l_i$, avec $i \neq j$ et, $l_i \in \mathcal{L}(\mathcal{P}_i)$ et $l_j \in \mathcal{L}(\mathcal{P}_j)$ sont des littéraux. D'un point de vue pratique, un mapping $\mathbf{K}(l_i) \Rightarrow l_j$ signifie que si $\mathcal{P}_i$ sait que l_i est vrai alors il doit notifier $\mathcal{P}_j$ que l_j est vrai. Un tel mapping définit quatre littéraux *partagés* par $\mathcal{P}_i$ et un pair $\mathcal{P}_j$: l_i et l_j et leur complément respectif $\bar{l}_i$ et $\bar{l}_j$. Notons qu'un mapping plus complexe de la forme $\mathbf{K}(f_i) \Rightarrow f_j$, où f_i et f_j sont des formules arbitraires de la logique propositionnelle respectivement en terme de $\mathcal{A}(\mathcal{P}_i)$ et $\mathcal{A}(\mathcal{P}_j)$, peut être simulé par un mapping tel que défini ci-dessus. Pour cela, il suffit d'utiliser deux nouvelles variables v_i et v_j, d'ajouter la forme clausale de $v_i \Leftrightarrow f_i$ à $\mathcal{T}(\mathcal{P}_i)$, d'ajouter la forme clausale de $v_j \Leftrightarrow f_j$ à $\mathcal{T}(\mathcal{P}_j)$, et enfin de poser le mapping $\mathbf{K}(v_i) \Rightarrow v_j$.

La *théorie globale* du P2PIS $\mathcal{S}$, notée $\mathcal{T}(\mathcal{S})$, est l'union des théories de ses pairs. Son *langage*, noté $\mathcal{L}(\mathcal{S})$, est l'ensemble fini de clauses sans répétition de littéral en terme de son *alphabet*, noté $\mathcal{A}(\mathcal{S})$, ce dernier étant l'union des alphabets de ses pairs. Son *langage cible*, noté $cible(\mathcal{L}(\mathcal{S}))$, est l'ensemble fini de clauses sans répétition de littéral en terme de son *alphabet cible*, noté $cible(\mathcal{A}(\mathcal{S}))$, celui-ci correspondant à l'union des alphabets cibles de ses pairs. Ici aussi, nous supposons que la clause vide $\square$ appartient à $\mathcal{L}(\mathcal{S})$ et $cible(\mathcal{L}(\mathcal{S}))$.
L'ensemble des *mappings* du P2PIS $\mathcal{S}$, noté $\mathcal{M}(\mathcal{S})$, est l'union des mappings de ses pairs.

Comme indiqué dans l'introduction, la particularité d'un P2PIS $\mathcal{S}$ est qu'aucun de ses pairs ne connait $\mathcal{T}(\mathcal{S})$, $\mathcal{L}(\mathcal{S})$, $\mathcal{A}(\mathcal{S})$, $cible(\mathcal{L}(\mathcal{S}))$, $cible(\mathcal{A}(\mathcal{S}))$ ou encore $\mathcal{M}(\mathcal{S})$.

Sémantique La sémantique d'un P2PIS est – bien évidemment – celle d'une théorie de la logique épistémique propositionnelle.

Notre objectif est maintenant de montrer que la définition d'un P2PIS modélise bien un système avec des mappings orientés entre les pairs. Plus précisément, nous montrons qu'un mapping allant d'un pair vers un autre spécifie bien un ensemble de connaissances que le premier pair doit observer, ainsi que les connaissances qu'il doit notifier au second pair si les connaissances observées sont satisfaites.
Pour cela, nous définissons la *saturation propositionnelle d'un P2PIS* qui enrichit de façon incrémentale les connaissances initiales des pairs à l'aide des mappings, lorsque les connaissances observées sont satisfaites. Nous montrons alors que le P2PIS et sa saturation sont équivalents par rapport aux conséquences dans le langage du P2PIS ($\mathcal{T}(\mathcal{S})\cup\mathcal{M}(\mathcal{S}) \models c$ ssi $PL(\mathcal{S}) \models c$ dans la Proposition 1). Nous exhibons aussi une caractérisation plus forte de ces conséquences : elle sont satisfaites par toutes les interprétations propositionnelles de tous les modèles épistémiques du P2PIS ($\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ ssi $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \mathbf{K}(c)$ dans la Proposition 1).

Définition 2 (Saturation propositionnelle d'un P2PIS) *Soit $\mathcal{S}$ un P2PIS. Sa* saturation propositionnelle $PL(\mathcal{S})$ *est le plus petit point fixe de :*
- $PL^0(\mathcal{S}) = \mathcal{T}(\mathcal{S})$
- $PL^{i+1}(\mathcal{S}) = PL^i(\mathcal{S}) \cup \{l_j \mid \mathbf{K}(l_k) \Rightarrow l_j \in \mathcal{M}(\mathcal{S})$ *et* $PL^i(\mathcal{S}) \models l_k\}$.

On remarquera qu'il existe toujours une saturation propositionnelle d'un P2PIS (c.-à-d. un plus petit point fixe pour la saturation) car le nombre de mappings d'un P2PIS est fini.

Proposition 1 *Soit $\mathcal{S}$ un P2PIS. Étant donnée une clause $c \in \mathcal{L}(\mathcal{S})$, $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ ssi $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \mathbf{K}(c)$ ssi $PL(\mathcal{S}) \models c$.*

Preuve. Nous devons montrer que $M = \{I \mid (I,\mathcal{W}) \models \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})\}$ est égal à $M' = \{J \in \mathcal{W} \mid (I,\mathcal{W}) \models \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})\}$, et est égal à $M'' = \{I \mid I \models PL(\mathcal{S})\}$.
Commençons par montrer que $PL(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \equiv \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$. Par définition de $PL(\mathcal{S})$, nous avons $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) = PL^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$. De plus, $PL^0(\mathcal{S}) \equiv PL'^0(\mathcal{S})$ avec $PL'^0(\mathcal{S}) = PL^0(\mathcal{S}) \cup \{l_i \mid PL^0(\mathcal{S}) \models l_i$ et $\mathbf{K}(l_i) \Rightarrow l_j \in \mathcal{M}(\mathcal{S})\}$. Nous avons alors $PL^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \equiv PL'^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$, d'où $PL'^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \mathbf{K}(l_i)$ car $l_i \in PL'^0(\mathcal{S})$, et donc $PL^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \mathbf{K}(l_i)$. Il découle alors que tout l_j tel que $PL^0(\mathcal{S}) \models l_i$ avec $\mathbf{K}(l_i) \Rightarrow l_j \in \mathcal{M}(\mathcal{S})$ est satisfait par toutes les interprétations de tout modèle épistémique de $PL^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ car $\mathbf{K}(l_i) \Rightarrow l_j \in \mathcal{M}(\mathcal{S})$. Par conséquent $PL^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \equiv PL^1(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ puisque $PL^1(\mathcal{S}) = PL^0(\mathcal{S}) \cup \{l_j \mid PL^0(\mathcal{S}) \models l_i$ et $\mathbf{K}(l_i) \Rightarrow l_j \in \mathcal{M}(\mathcal{S})\}$. En itérant ce raisonnement jusqu'au point fixe $PL^n(\mathcal{S})$ (qui existe toujours), nous avons $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) = PL^0(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \equiv \cdots \equiv PL^n(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) = PL(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$.
Puisque nous avons $PL(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \equiv \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$, nous avons aussi, par définition d'un modèle épistémique $(I,\mathcal{W})$ de $PL(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$, que tout $J \in \mathcal{W}$ est tel que $J \models PL(\mathcal{S})$, donc $I \models PL(\mathcal{S})$, c.-à-d. $M \subseteq M' \subseteq M''$.
Montrons maintenant que $M'' \subseteq M \subseteq M'$. Soit $(I,\mathcal{W})$ une interprétation épistémique telle que $\mathcal{W} = \{J \mid J \models PL(\mathcal{S})\}$. Notons que $(I,\mathcal{W}) \models \mathcal{T}(\mathcal{S})$ puisque $(I,\mathcal{W}) \models PL(\mathcal{S})$ et $\mathcal{T}(\mathcal{S}) \subseteq PL(\mathcal{S})$. Si $(I,\mathcal{W}) \not\models \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ alors il existe $\mathbf{K}(l_i) \Rightarrow l_j \in \mathcal{M}(\mathcal{S})$ tel que $(I,\mathcal{W}) \not\models \mathbf{K}(l_i) \Rightarrow l_j$, c.-à-d. $(I,\mathcal{W}) \models \mathbf{K}(l_i)$ et $(I,\mathcal{W}) \not\models l_j$. Ceci mène à une contradiction car si $(I,\mathcal{W}) \models \mathbf{K}(l_i)$ alors pour tout $J \in \mathcal{W}$ $J \models l_i$ et par conséquent $PL(\mathcal{S}) \models l_i$. Il en découle, par construction de $PL(\mathcal{S})$, que $l_j \in PL(\mathcal{S})$, c.-à-d. $(I,\mathcal{W}) \models l_j$. Ainsi, toute interprétation épistémique $(I,\mathcal{W})$ telle que $\mathcal{W} = \{J \mid J \models PL(\mathcal{S})\}$ est un modèle épistémique de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$. ■

Pour conclure cette section, nous présentons le P2PIS "jouet" que nous utiliserons ultérieurement dans cet article afin d'illustrer le comportement de notre algorithme de calcul de conséquences.

Exemple La Figure 1 représente le P2PIS $\mathcal{S}$ composé des pairs $\mathcal{P}_1$ et $\mathcal{P}_2$. Les théories des pairs sont les nœuds étiquetés par les noms des pairs et les mappings entre les pairs étiquettent l'arête qui lie leur théorie respective.
Concernant les théories des pairs, nous supposons ici que toutes les variables sont cibles, à l'exception de C_1 sur $\mathcal{P}_1$ et de F_2 sur $\mathcal{P}_2$.
Quant aux mappings qui connectent les pairs, on remarquera que $\mathbf{K}(C_1) \Rightarrow E_2$ est orienté de $\mathcal{P}_1$ vers $\mathcal{P}_2$ alors que les quatre autres mappings sont orientés dans le sens inverse.
Pour finir, nous rappelons la signification d'un mapping en prenant $\mathbf{K}(C_1) \Rightarrow E_2$ pour exemple : si $\mathcal{P}_1$ sait que C_1 est vrai, alors il notifie $\mathcal{P}_2$ que E_2 est vrai.

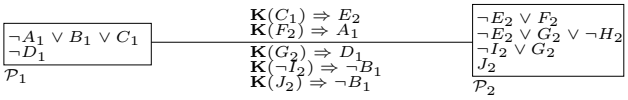

FIG. 1 – Le P2PIS $\mathcal{S}$.

4 Le calcul de conséquences

Nous étudions maintenant le problème de trouver les conséquences les plus fortes résultant d'une clause soumise à un pair d'un P2PIS. Plus formellement, nous voulons trouver les *impliqués clausaux premiers propres d'une clause exprimée dans le langage d'un pair d'un P2PIS, par rapport aux connaissances de ce P2PIS et dans le langage cible de ce P2PIS*.
Nous définissons tout d'abord ces impliqués, puis nous donnons leur complexité. Enfin, nous posons le problème de calcul de conséquences pour lequel nous fournissons l'algorithme DECA$_{\mathbf{K}}$ dans la prochaine section.

Définition 3 (Impliqué clausal premier propre) *Soient $\mathcal{T}$ une théorie de la logique épistémique propositionnelle constituée de formules en terme d'un alphabet $\mathcal{A}$ et, f et g deux clauses en terme de $\mathcal{A}$.*

- *f est un* impliqué clausal de g par rapport à $\mathcal{T}$ *ssi* $\mathcal{T} \cup \{g\} \models f$.
- *f est un* impliqué clausal premier de g par rapport à $\mathcal{T}$ *ssi f est un impliqué clausal de g par rapport à $\mathcal{T}$ et pour toute clause f' en terme de $\mathcal{A}$, si $\mathcal{T} \cup \{g\} \models f'$ et $f' \models f$ alors $f \models f'$ ($f' \equiv f$).*
- *f est un* impliqué clausal propre de g par rapport à $\mathcal{T}$ *ssi f est un impliqué clausal de g par rapport à $\mathcal{T}$ mais $\mathcal{T} \not\models f$.*
- *f est un* impliqué clausal premier propre de g par rapport à $\mathcal{T}$ *ssi f est à la fois un impliqué clausal premier et propre de g par rapport à $\mathcal{T}$.*

Le théorème suivant fait le lien entre la notion d'impliqué clausal premier propre et la classe de complexité BH_2 qui regroupe les problèmes de décision aussi durs, mais pas plus durs, qu'un problème SAT *et* un problème UNSAT en logique propositionnelle. Son problème canonique est d'ailleurs le problème SAT–UNSAT en logique propositionnelle, dont une instance SAT–UNSAT(ϕ_1, ϕ_2) est vraie si et seulement si ϕ_1 est satisfiable et ϕ_2 est insatisfiable, où ϕ_1, ϕ_2 sont des formules de la logique propositionnelle.

Théorème 1 (Complexité des impliqués de la Définition 3) *Soient $\mathcal{T}$ une théorie de la logique épistémique propositionnelle constituée de formules en terme d'un alphabet propositionnel $\mathcal{A}$ et, f et g deux clauses en terme de $\mathcal{A}$. Décider si f est un impliqué clausal premier propre de g par rapport à $\mathcal{T}$ est BH_2-complet.*

Preuve. L'appartenance à BH_2-difficile vient directement du fait que ce problème est BH_2-difficile dans le cas d'une théorie $\mathcal{T}$ de la logique propositionnelle (*cf.* [23]).
En reprenant le même argumentaire que pour l'appartenance du problème à BH_2-facile dans le cas d'une théorie $\mathcal{T}$ de logique propositionnelle (*cf.* [23]), nous montrons que le problème est un problème SAT–UNSAT de la logique épistemique propositionnelle. En effet, décider si une clause f est un impliqué premier propre d'une clause g par rapport à une théorie $\mathcal{T}$ de la logique épistémique propositionnelle revient à :

- décider si f est un impliqué de $\mathcal{T} \cup \{g\}$, c.-à-d. décider $\mathcal{T} \cup \{g\} \models f$, d'où $\mathcal{T} \cup \{g, \neg f\} \models \Box$, soit résoudre un problème UNSAT,
- décider si f est un impliqué premier de $\mathcal{T} \cup \{g\}$, c.-à-d. décider si $\mathcal{T} \cup \{g\} \not\models f'$, d'où $\mathcal{T} \cup \{g, \neg f'\} \not\models \Box$, pour tout f' correspondant à f privé d'un de ses littéraux, soit N problèmes SAT si f a N littéraux
- et enfin décider si f est un impliqué propre de $\mathcal{T}$, c.-à-d. décider $\mathcal{T} \not\models f$, d'où $\mathcal{T} \cup \{\neg f\} \not\models \Box$, soit encore un problème SAT.

Puisque k instances du problème SAT (ici $k = N + 1$) peuvent être résolues en une instance SAT en s'assurant par renommage que ces k instances sont rendues indépendantes, décider si f est un impliqué premier propre de g par rapport à $\mathcal{T}$ peut être ici réduit en temps polynomial à une instance de SAT-UNSAT en logique épistémique propositionnelle.
SAT (respectivement UNSAT) est NP-complet (respectivement coNP-complet) en logique propositionnelle (*cf.* [15]) et en logique épistemique propositionnelle (*cf.* [22, 18]). Notre problème est par conséquent un problème SAT–UNSAT en logique propositionnelle (à une réduction polynomiale près), le problème canonique de BH_2. ■

Il est intéressant d'exhiber les liens existant entre les impliqués clausaux premiers propres dans les P2PIS définis ci-dessus et dans ceux de [1, 2, 3] qui utilisent des *clauses* pour modéliser des mappings non orientés.
D'un point de vue sémantique, tout P2PIS de [1, 2, 3] peut être transformé de façon équivalente (en introduisant éventuellement de nouvelles variables) en un P2PIS avec des mappings de la forme $l_i \Rightarrow l_j$. Par conséquent, passer d'un P2PIS avec des mappings orientés à des mappings non orientés, et vice-versa, revient à ajouter/supprimer l'opérateur épistémique $\mathbf{K}$ dans les mappings. S'appuyant sur ce fait, la Proposition 2 établit qu'un impliqué clausal premier propre d'une clause soumise à un pair par rapport à un P2PIS avec des mappings orientés est *seulement* un impliqué clausal de cette même clause par rapport au même P2PIS avec des mappings non orientés. Il découle de cette proposition que les deux approches ne sont – bien évidemment – pas équivalentes.
Toutefois, les approches sont de même complexité (BH_2-complet) en ce qui concerne les impliqués clausaux premiers propres (*cf.* Théorème 1 et [3]).

Proposition 2 *Soient $\mathcal{S}$ un P2PIS avec mappings orientés, dont l'un des pairs est $\mathcal{P}$, et $\mathcal{S}'$ le P2PIS obtenu en remplaçant tout mapping de la forme $\mathbf{K}(l_i) \Rightarrow l_j$ par $l_i \Rightarrow l_j$ dans $\mathcal{S}$. Tout impliqué clausal premier propre d'une clause $c \in \mathcal{L}(\mathcal{P})$ par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ est un impliqué clausal (non nécessairement premier ou propre) de c par rapport à $\mathcal{T}(\mathcal{S}') \cup \mathcal{M}(\mathcal{S}')$. Tout impliqué clausal (premier propre) d'une clause $c \in \mathcal{L}(\mathcal{P})$ par rapport à $\mathcal{T}(\mathcal{S}') \cup \mathcal{M}(\mathcal{S}')$ n'est pas nécessairement un impliqué clausal de c par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$.*

Preuve. Si $M = \{I \mid (I, \mathcal{W}) \models \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})[\cup\{c\}]\}$ et $M' = \{I \mid I \models \mathcal{T}(\mathcal{S}') \cup \mathcal{M}(\mathcal{S}')[\cup\{c\}]\}$ alors $M' \subseteq M$. Ceci est vrai car $I \models \mathcal{T}(\mathcal{S}') \cup \mathcal{M}(\mathcal{S}')$ ssi $(I, \{I\}) \models \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$. De plus, dans le cas général, nous avons $M \neq M'$. Par exemple, A_1 est une conséquence du P2PIS ci-dessous avec le mapping $B_1 \Rightarrow C_2$, alors ce que c'est pas le cas avec le mapping $\mathbf{K}(B_1) \Rightarrow C_2$: 

Vérifier si une clause est une conséquence de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})[\cup\{c\}]$ (respectivement $\mathcal{T}(\mathcal{S}') \cup \mathcal{M}(\mathcal{S}')[\cup\{c\}]$) est décidé par rapport à M (respectivement M'). Par conséquent, la Proposition 2 résulte de la Définition 3, de $M' \subseteq M$ et de ce qu'en général $M' \neq M$. ■

Enfin, nous concluons cette section en définissant le problème de calcul de conséquences pour lequel nous fournissons l'algorithme DECA$_{\mathbf{K}}$ dans la section suivante.

Définition 4 (Problème de calcul de conséquences) *Soit $\mathcal{S}$ un P2PIS. Étant donnés un pair $\mathcal{P}_k$ de $\mathcal{S}$ et une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, le* problème de calcul de conséquences *est de trouver l'ensemble de tous les impliqués clausaux premiers propres de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$, et dans le langage cible de $\mathcal{S}$.*

5 L'algorithme de calcul de conséquences : DECA$_{\mathbf{K}}$

DECA$_{\mathbf{K}}$ (*cf.* Algorithme 1), pour DEcentralized Consequence finding Algorithm, fournit une solution au problème de calcul de conséquences de la Définition 4. Nous commençons par introduire le principe de fonctionnement de DECA$_{\mathbf{K}}$, avant de l'illustrer sur l'exemple "jouet" présenté précédemment (*cf.* Fig. 1). Enfin, nous établissons les propriétés de DECA$_{\mathbf{K}}$.

Description de DECA$_{\mathbf{K}}$ **:** DECA$_{\mathbf{K}}$ est un algorithme récursif décentralisé qui s'exécute sur chaque pair d'un P2PIS. Un point clé de cet algorithme est l'utilisation d'un historique, $hist$ dans l'Algorithme 1.
Un premier rôle de cet historique est de mémoriser toutes les clauses pour lesquelles des pairs sont sollicités lors des appels récursifs d'un calcul de conséquences : ces clauses sont des conséquences du P2PIS et de la clause dont DECA$_{\mathbf{K}}$ est en train de calculer les conséquences. L'idée sous-jacente est que lorsqu'un pair est sollicité plusieurs fois durant le calcul des conséquences d'une clause, il doit se rappeler à chaque fois de toutes les clauses pour lesquelles il a déjà été sollicité, c.-à-d. les clauses de l'historique qui sont dans *son* langage. En effet, il doit les utiliser dans son raisonnement en plus de ses propres connaissances (c.-à-d. sa théorie et ses mappings) car ces clauses, bien qu'elles soient dans son langage et satisfaites par le système global, ne peuvent pas être déduites à partir des seules connaissances de ce pair.
Un second rôle de l'historique est de pouvoir garantir la terminaison de DECA$_{\mathbf{K}}$. En effet, l'historique permet de stopper les cycles de raisonnement car ils sont repérables lorsqu'un pair est sollicité deux fois avec la même clause au cours d'un même calcul.

Nous donnons maintenant un aperçu de la façon dont DECA$_{\mathbf{K}}$ effectue un calcul de conséquences.
Étant donnée une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$ soumise au pair $\mathcal{P}_k$, DECA$_{\mathbf{K}}{}^k$ (DECA$_{\mathbf{K}}$ sur $\mathcal{P}_k$) commence par calculer des impliqués clausaux de q_k par rapport à $\mathcal{T}(\mathcal{P}_k) \cup hist$ à la ligne 4, incluant tous les impliqués clausaux premiers propres puisque la résolution linéaire est une procédure qui termine, qui est correcte pour le calcul d'impliqués clausaux, et qui est complète pour le calcul d'impliqués clausaux premiers propres[1] (*cf.* [24]).
Dans une seconde étape, lignes 9 à 11, DECA$_{\mathbf{K}}{}^k$ calcule les impliqués clausaux de q_k par rapport à $\mathcal{T}(\mathcal{P}_k) \cup hist$ *et les connaissances qui sont notifiées à $\mathcal{P}_k$ par les mappings*, incluant tous les impliqués clausaux premiers propres. Pour cela, DECA$_{\mathbf{K}}{}^k$ supprime chaque littéral partagé l_k dans impliqués obtenus à l'étape précédente, si $\bar{l}_k$ est notifié à $\mathcal{P}_k$ grâce à un mapping $\mathbf{K}(l_j) \Rightarrow \bar{l}_k$. Afin de décider si la notification est faite ou non, DECA$_{\mathbf{K}}{}^k$ sollicite DECA$_{\mathbf{K}}{}^j$ avec $\bar{l}_j$ afin de tester si une inconsistance apparait.
Dans une dernière étape, lignes 15 à 17, DECA$_{\mathbf{K}}{}^k$ considère chaque littéral l_k qui est un impliqué clausal de q_k résultant de l'étape précédente, tel qu'il existe un mapping $\mathbf{K}(l_k) \Rightarrow l_j$. Pour chacun de ces littéraux, DECA$_{\mathbf{K}}{}^k$ sollicite DECA$_{\mathbf{K}}{}^j$ pour calculer tous les *impliqués clausaux premiers propres* de l_j par rapport au P2PIS.
Évidemment, DECA$_{\mathbf{K}}{}^k$ utilise les conditions d'arrêt appropriées pour garantir la terminaison en présence de raisonnement cyclique (ligne 2) et pour éviter des calculs non nécessaires (lignes 6 et 13), ainsi que le filtrage adéquat pour ne retourner que des clauses dans le langage cible du P2PIS (lignes 8 et 18).

On notera que la variable booléenne $forth$ de DECA$_{\mathbf{K}}{}^k$ permet de distinguer les appels (récursifs) de la seconde étape ($forth = false$), de ceux de la troisième étape ($forth = true$) : les premiers utilisent des mappings $\mathbf{K}(l_m) \Rightarrow l_n$ pour aller de $\mathcal{P}_n$ à $\mathcal{P}_m$ afin de vérifier si $\mathcal{P}_m$ notifie $\mathcal{P}_n$ de l_n, alors que les seconds utilisent des mappings $\mathbf{K}(l_m) \Rightarrow l_n$ pour aller de $\mathcal{P}_m$ à $\mathcal{P}_n$ afin de calculer des impliqués clausaux de l_m.
Plus précisément, le booléen $forth$ empêche d'imbriquer un appel récursif de la troisième étape (ligne 17) dans un appel récursif de la seconde (ligne 10). En effet, pour décider si via un mapping $\mathbf{K}(l_j) \Rightarrow \bar{l}_k$ le pair $\mathcal{P}_j$ notifie $\mathcal{P}_k$ que $\bar{l}_k$ est vrai, DECA$_{\mathbf{K}}{}^k$ *suppose* que $\bar{l}_j$ est vrai en sollicitant DECA$_{\mathbf{K}}{}^j$. Si celui-ci retourne la clause vide, c'est qu'il a utilisé $\bar{l}_j$ avec une clause de la forme $l_j \vee l_j^1 \vee \cdots \vee l_j^n$, et a montré, éventuellement à l'aide d'appels récursifs, que $l_j^1 \vee \cdots \vee l_j^n \equiv \Box$ dans le P2PIS, c.-à-d. que l_j est vrai dans le P2PIS, et donc que $\bar{l}_k$ est vrai dans le P2PIS. Il est important de noter qu'effectuer un appel récursif de la troisième étape (ligne 17) dans un appel récursif de la seconde (ligne 10) serait incorrect. Cela signifierait que DECA$_{\mathbf{K}}{}^j$ a utilisé $\bar{l}_j$ avec une clause de la forme $l_j \vee l_j^1 \vee \cdots \vee l_j^n$ et qu'il a montré que $l_j^1 \vee \cdots \vee l_j^n \equiv l_j^q$, $1 \leq q \leq n$, afin d'utiliser un mapping $\mathbf{K}(l_j^q) \Rightarrow l_p$. Or ce n'est pas nécessairement vrai car DECA$_{\mathbf{K}}{}^k$ avait seulement *supposé* que $\bar{l}_j$ était vrai. Ce qu'a donc réellement montré DECA$_{\mathbf{K}}{}^j$ est que $l_j \vee l_j^1 \vee \cdots \vee l_j^n \equiv l_j \vee l_j^q$, ce qui n'est pas suffisant pour utiliser le mapping $\mathbf{K}(l_j^q) \Rightarrow l_p$!

[1] Par abus de langage, nous simplifions ici et dans la suite de l'article en assimilant la complétude de la résolution linéaire pour le calcul des impliqués clausaux premiers propres, à l'existance de certaines stratégies de résolution linéraire complètes pour ce calcul (*cf.* [24]).

Algorithme 1: DECA$_{\mathbf{K}}$ du pair $\mathcal{P}_k$ dans un P2PIS $\mathcal{S}$

DECA$_{\mathbf{K}}{}^k(q_k)$

Entrée: une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$

Sortie: un ensemble d'impliqués clausaux de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ qui appartiennent à $cible(\mathcal{L}(\mathcal{S}))$, incluant tous les premiers propres.

(1) **return** DECA$_{\mathbf{K}}{}^k(q_k, \emptyset, true)$

DECA$_{\mathbf{K}}{}^k(q_k, hist, forth)$

Entrée: une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, un ensemble de clauses $hist$, un booléen $forth$

Sortie: un ensemble d'impliqués clausaux de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ qui appartiennent à $cible(\mathcal{L}(\mathcal{S}))$, incluant tous les premiers propres.

(1) **if** q_k a déjà été traité (c.-à-d. $q_k \in hist$)
(2) **return** $\emptyset$ % plus d'impliqué propre (cycle)
(3) **else**
(4) $CI \leftarrow \{q_k\} \cup ResolutionLineaire(q_k, \mathcal{T}(\mathcal{P}_k) \cup hist)$
(5) **if** $\Box \in CI$
(6) **return** $\{\Box\}$ % plus d'impliqué premier ($\Box$ est premier)
(7) **else**
(8) retirer de CI toute clause contenant un littéral qui n'est pas dans $cible(\mathcal{L}(\mathcal{P}_k))$ et n'est pas partagé
(9) **foreach** littéral l_k apparaissant dans des clauses de CI
(10) **if** $\mathbf{K}(l_j) \Rightarrow \bar{l}_k \in \mathcal{M}(\mathcal{P}_k)$ et $\Box \in$ DECA$_{\mathbf{K}}{}^j(\bar{l}_j, hist \cup \{q_k\}, false)$
(11) retirer toutes les occurrences de l_k des clauses de CI
(12) **if** $\Box \in CI$
(13) **return** $\{\Box\}$ % plus d'impliqué premier ($\Box$ est premier)
(14) **if** $forth = true$
(15) **foreach** clause de taille "un" l_k (c.-à-d. a littéral) de CI
(16) **if** $\mathbf{K}(l_k) \Rightarrow l_j \in \mathcal{M}(\mathcal{P}_k)$
(17) ajouter DECA$_{\mathbf{K}}{}^j(l_j, hist \cup \{q_k\}, true)$ à CI
(18) retirer de CI toute clause contenant un littéral qui est dans $\mathcal{L}(\mathcal{P}_k)$ mais pas dans $cible(\mathcal{L}(\mathcal{P}_k))$
(19) **return** CI

Exemple (suite) Supposons que nous voulons obtenir les impliqués clausaux premiers propres de A_1 depuis le pair $\mathcal{P}_1$ du P2PIS de la Figure 1. Nous faisons l'appel DECA$_{\mathbf{K}}{}^1(A_1)$ qui déclenche à son tour l'appel DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$.

ÉTAPE 1 DE DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$.

A_1 et $B_1 \vee C_1$ sont des impliqués clausaux de A_1 par rapport à $\mathcal{T}(\mathcal{P}_1)$, obtenus par résolution linéaire à la ligne 4. A_1 et $B_1 \vee C_1$ restent dans CI à la ligne 8, car ils ne contiennent pas de littéral non partagé n'appartenant pas à $cible(\mathcal{L}(\mathcal{P}_1))$.

ÉTAPE 2 DE DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$.

Du fait des mappings $\mathbf{K}(\neg I_2) \Rightarrow \neg B_1$ et $\mathbf{K}(J_2) \Rightarrow \neg B_1$, DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$ déclenche deux appels récursifs à la ligne 10 : DECA$_{\mathbf{K}}{}^2(I_2, \{A_1\}, false)$ et DECA$_{\mathbf{K}}{}^2(\neg J_2, \{A_1\}, false)$. Le dernier retournant $\{\Box\}$, $\neg B_1$ est vrai dans le P2PIS.

• DECA$_{\mathbf{K}}{}^2(I_2, \{A_1\}, false)$ trouve I_2 et G_2 (ligne 4) comme impliqués clausaux de I_2 par rapport à $\mathcal{T}(\mathcal{P}_2) \cup \{A_1\}$. Il n'y a pas d'appel récursif à la ligne 10 puisque ni $\neg I_2$ ni $\neg G_2$ apparait à droite de $\Rightarrow$ dans un mapping. Il n'y a pas d'appel récursif à la ligne 17 car $forth = false$. Par conséquent, on ne peut savoir si $\neg B_1$ est vrai dans $\mathcal{P}_1$ via $\mathbf{K}(\neg I_2) \Rightarrow \neg B_1$.

• DECA$_{\mathbf{K}}{}^2(\neg J_2, \{A_1\}, false)$ trouve $\neg J_2$ et $\Box$ (ligne 4) comme impliqués clausaux de $\neg J_2$ par rapport à $\mathcal{T}(\mathcal{P}_2) \cup \{A_1\}$. DECA$_{\mathbf{K}}{}^2(\neg J_2, \{A_1\}, false)$ s'arrête donc en retournant $\{\Box\}$ à la ligne 6. Ici, le mapping $\mathbf{K}(J_2) \Rightarrow \neg B_1$ permet de déduire que $\neg B_1$ est vrai dans $\mathcal{P}_1$.

Il découle des deux appels récursifs ci-dessus que $CI = \{A_1, C_1\}$ après la ligne 11 de DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$.

ÉTAPE 3 DE DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$.

Avec $forth = true$ et le mapping $\mathbf{K}(C_1) \Rightarrow E_2$, DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$ déclenche l'appel récursif DECA$_{\mathbf{K}}{}^2(E_2, \{A_1\}, true)$ qui retourne $\{E_2, G_2 \vee \neg H_2\}$ à la ligne 17.

• DECA$_{\mathbf{K}}{}^2(E_2, \{A_1\}, true)$ trouve E_2, F_2 et $G_2 \vee \neg H_2$ (ligne 4) comme impliqués clausaux de E_2 par rapport à $\mathcal{T}(\mathcal{P}_2) \cup \{A_1\}$. À la ligne 8, ces derniers restent dans CI car ils ne contiennent pas de littéraux non partagés n'appartenant pas à $cible(\mathcal{L}(\mathcal{P}_2))$. Il n'y a pas d'appel récursif à la ligne 10 car ni $\neg E_2$, ni $\neg F_2$, ni $\neg G_2$, ni H_2 n'apparait à droite de $\Rightarrow$ dans un mapping.

L'appel récursif DECA$_{\mathbf{K}}{}^1(A_1, \{E_2, A_1\}, true)$ est déclenché à la ligne 17 du fait de $forth = true$ et du mapping $\mathbf{K}(F_2) \Rightarrow A_1$. Cet appel s'arrête à la ligne 2 car l'historique indique que A_1 a déjà été traité, donc A_1 est satisfait et par conséquent il n'y a pas d'impliqué propre de A_1 par rapport à $\mathcal{T}(\mathcal{P}_1) \cup hist$. À la ligne 18 de DECA$_{\mathbf{K}}{}^2(E_2, \{A_1\}, true)$, nous avons $CI = \{E_2, G_2 \vee \neg H_2\}$ car F_2 n'appartient pas à $cible(\mathcal{L}(\mathcal{P}_2))$.

Il découle de l'appel récursif ci-dessus que DECA$_{\mathbf{K}}{}^1(A_1, \emptyset, true)$ retire C_1 de CI à la ligne 18 et retourne A_1, E_2 et $G_2 \vee \neg H_2$ comme impliqués clausaux de A_1 par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$. On notera que ce sont *tous* les impliqués clausaux premiers propres de A_1 par rapport au P2PIS $\mathcal{S}$ et appartenant à $cible(\mathcal{L}(\mathcal{S}))$.

Pour conclure cet exemple, mettons en évidence le résultat de la Proposition 2 afin d'illustrer qu'orienter ou non les mappings a une influence sur le calcul d'impliqués premiers propres. Pou cela, considérons le P2PIS $\mathcal{S}'$ obtenu en remplaçant tout mapping $\mathbf{K}(l_i) \Rightarrow l_j$ de $\mathcal{S}$ par $l_i \Rightarrow l_j$. On peut facilement vérifier, par exemple par résolution linéaire de A_1 avec $\mathcal{T}(\mathcal{S}') \cup \mathcal{M}(\mathcal{S}')$, que A_1, E_2 et $\neg H_2$ sont *tous* les impliqués clausaux premiers propres de A_1 par rapport au P2PIS $\mathcal{S}'$ et appartenant à $cible(\mathcal{L}(\mathcal{S}'))$. Nous avons donc bien que les impliqués clausaux premiers

propres obtenus avec les mappings orientés sont seulement des impliqués clausaux avec les mappings non orientés (cas de $G_2 \vee \neg H_2$), alors que les impliqués clausaux premiers propres obtenus avec les mappings non orientés ne sont pas nécessairement des impliqués clausaux avec les mappings orientés (cas de $\neg H_2$).

Propriétés de $\text{DECA}_{\mathbf{K}}$ Nous montrons maintenant que $\text{DECA}_{\mathbf{K}}$ *termine* (Théorème 2), qu'il est *correct* pour le calcul d'impliqués clausaux d'une clause dans le langage d'un pair par rapport à un P2PIS et dans le langage de ce P2PIS, et qu'il est *complet* par rapport au problème de la Définition 4 (Théorème 3). On remarquera que les propriétés de $\text{DECA}_{\mathbf{K}}$ restent vraies même dans le cas de P2PIS *insatisfiables*.

Théorème 2 (Terminaison de $\text{DECA}_{\mathbf{K}}$**)** *Soit $\mathcal{S}$ un P2PIS dont l'un des pairs est $\mathcal{P}_k$. Étant donnée une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, $\text{DECA}_{\mathbf{K}}{}^k(q_k)$ s'arrête après un temps fini.*

Preuve. Si $\text{DECA}_{\mathbf{K}}$ ne s'arrêtait pas, il y aurait un nombre infini d'appels récursifs, et donc un historique infini. Ceci est impossible car $\text{DECA}_{\mathbf{K}}$ s'arrête grâce à l'historique quand une même clause est traitée deux fois (ligne 2), et le nombre de clauses possibles à traiter est fini puisque le nombre de mappings est fini. ■

Théorème 3 (Correction et complétude de $\text{DECA}_{\mathbf{K}}$**)** *Soit $\mathcal{S}$ un P2PIS dont l'un des pairs est $\mathcal{P}_k$. Étant donnée une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$:*
- *toute clause de $\text{DECA}_{\mathbf{K}}{}^k(q_k)$ est un impliqué clausal de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ et appartenant à $cible(\mathcal{L}(\mathcal{S}))$,*
- *si $i \in cible(\mathcal{L}(\mathcal{S}))$ est un impliqué clausal premier propre de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ alors $i \in \text{DECA}_{\mathbf{K}}{}^k(q_k)$.*

Preuve. Tout d'abord, il est important de remarquer que la correction et la complétude de $\text{DECA}_{\mathbf{K}}$ sont toujours vérifiées dans un P2PIS *insatisfiable* puisque toute clause est un *impliqué* de n'importe quelle autre clause par rapport à un P2PIS insatisfiable et il n'existe pas d'impliqué clausal *propre* d'une clause par rapport à un P2PIS insatisfiable.

Considérons maintenant des P2PIS *satisfiables*.
Rappel : la résolution linéaire (*cf.* [11, 21]) est correcte pour le calcul des impliqués clausaux d'une clause par rapport à une théorie de logique propositionnelle, et est complète pour le calcul de tous les impliqués clausaux premiers propres par rapport à une théorie de logique propositionnelle (*cf.* [24]).

La preuve de correction est faite par récurrence sur le nombre d'appels à $\text{DECA}_{\mathbf{K}}$ avec $forth = true$ pour produire une clause de $\text{DECA}_{\mathbf{K}}{}^k(q_k)$.
Elle découle facilement de l'utilisation de la résolution linéaire, du Lemme 1 (ligne 4) et du Lemme 2 (ligne 11).

La preuve de complétude s'appuye sur le nombre n de pas de saturation nécessaire pour que $PL^n(\mathcal{S}') \models i$ alors que $PL^{n-1}(\mathcal{S}') \not\models i$ si $PL^{n-1}(\mathcal{S}')$ existe, où $i \in cible(\mathcal{L}(\mathcal{S}))$ est un impliqué clausal premier propre de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ et $PL(\mathcal{S}')$ est la saturation propositionnelle du P2PIS $\mathcal{S}'$ avec $\mathcal{T}(\mathcal{S}') = \mathcal{T}(\mathcal{S}) \cup \{q_k\}$ et $\mathcal{M}(\mathcal{S}') = \mathcal{M}(\mathcal{S})$. La preuve est faite par récurrence sur n en montrant que s'il faut n pas de saturation pour obtenir i alors i est obtenu par $\text{DECA}_{\mathbf{K}}(q_k, \emptyset, true)$ avec un nombre d'appels récursifs avec $forth = true$ au plus égal à n. ■

Lemme 1 *Soit $\mathcal{S}$ un P2PIS satisfiable dont l'un des pairs est $\mathcal{P}_k$. Étant donnée une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, toute clause c obtenue par résolution linéraire de q_k avec $\mathcal{T}(\mathcal{P}_k)$ est un impliqué clausal de q_k par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$.*

Preuve. Toute clause c produite par résolution linéaire de q_k avec $\mathcal{T}(\mathcal{P}_k)$ est telle que $PL(\mathcal{S}') \models c$, où $PL(\mathcal{S}')$ est la saturation propositionnelle du P2PIS $\mathcal{S}'$ avec $\mathcal{T}(\mathcal{S}') = \mathcal{T}(\mathcal{S}) \cup \{q_k\}$ et $\mathcal{M}(\mathcal{S}') = \mathcal{M}(\mathcal{S})$. En utilisant la Proposition 1, nous obtenons $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \models c$. ■

Lemme 2 *Soit $\mathcal{S}$ un P2PIS satisfiable dont l'un des pairs est $\mathcal{P}_j$. Étant donné un littéral $l_j \in \mathcal{L}(\mathcal{P}_j)$: $\square \in \text{DECA}_{\mathbf{K}}{}^j(\bar{l}_j, \emptyset, false)$ ssi $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \mathbf{K}(l_j)$.*

Preuve. Nous prouvons ($\Rightarrow$) par récurrence sur le nombre n d'appels à $\text{DECA}_{\mathbf{K}}$.
Pour $n = 1$, $hist$ n'est pas nécessairement vide (car lorsque nous utiliserons l'hypothèse de récurrence, les appels récursifs qui arrêteront le raisonnement auront un historique non vide) et $\square$ est nécessairement un impliqué clausal de $\bar{l}_j$ (Lemme 1) obtenu à la ligne 6 à partir de $\mathcal{T}(\mathcal{P}_j) \cup hist$. En utilisant la Définition 2, nous avons $PL(\mathcal{S}') \models \mathcal{T}(\mathcal{P}_j) \cup hist$, avec $\mathcal{T}(\mathcal{S}') = \mathcal{T}(\mathcal{S}) \cup hist$ et $\mathcal{M}(\mathcal{S}') = \mathcal{M}(\mathcal{S})$. Il en découle que $PL(\mathcal{S}') \cup \{\bar{l}_j\} \models \square$, donc $PL(\mathcal{S}') \models l_j$ et en utilisant la Proposition 1 : $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \models \mathbf{K}(l_j)$.
Supposons maintenant que ($\Rightarrow$) est vrai pour $1 \leq n < \alpha$.
Pour $n = \alpha$, $\square$ est obtenue à la ligne 11. Notons que $\text{DECA}_{\mathbf{K}}$ s'arrête à la ligne 19 et tous les appels récursifs sont ceux de la ligne 10 puisque $forth = false$ interdit d'utiliser ceux de la ligne 17. Il existe donc une clause $l_j \vee l_j^1 \vee \cdots \vee l_j^p$ telle que $\mathcal{T}(\mathcal{P}_j) \cup hist \models l_j \vee l_j^1 \vee \cdots \vee l_j^p$ (ligne 4) et pour $i \in [1..p]$: $\mathbf{K}(l_i) \Rightarrow \bar{l}_j^i \in \mathcal{M}(\mathcal{S})$ et $\square \in \text{DECA}_{\mathbf{K}}{}^i(\bar{l}_i, \{\bar{l}_j\}, false)$ (ligne 11). Tout appel récursif déclenché par $\text{DECA}_{\mathbf{K}}{}^j(\bar{l}_j, \emptyset, false)$ déclenche à son tour moins de α appels récursifs. Par conséquent, pour tout $i \in [1..p]$ nous avons $PL(\mathcal{S}') \models \bar{l}_j^i$, avec $\mathcal{T}(\mathcal{S}') = \mathcal{T}(\mathcal{S}) \cup hist \cup \{\bar{l}_j\}$ et $\mathcal{M}(\mathcal{S}') = \mathcal{M}(\mathcal{S})$. En utilisant la Proposition 1, nous avons que pour tout $i \in [1..p]$ $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{\bar{l}_j\} \models \bar{l}_j^i$, et par conséquent $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{\bar{l}_j\} \models \square$, donc $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \models l_j$, et ainsi $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \models \mathbf{K}(l_j)$.

Nous prouvons maintenant ($\Leftarrow$) par récurrence sur le nombre n de pas qui mène à $PL^n(\mathcal{S}) \models l_j$, alors que $PL^{n-1}(\mathcal{S}) \not\models l_j$ si $PL^{n-1}(\mathcal{S})$ existe.
Pour $n = 0$, nous avons $\mathcal{T}(\mathcal{P}_j) \models l_j$ et, du fait de la ligne 4 de DECA$_{\mathbf{K}}$, $\Box \in$ DECA$_{\mathbf{K}}{}^j(\bar{l}_j, \emptyset, false)$. Notons que DECA$_{\mathbf{K}}$ ne peut pas s'arrêter à la ligne 2 car cela signifierait que $\bar{l}_j$ a déjà été traité, et donc que DECA$_{\mathbf{K}}$ se serait déjà arrêté à la ligne 6 puisque $\mathcal{T}(\mathcal{P}_j) \models l_j$.
Supposons maintenant que ($\Leftarrow$) est vrai pour $0 \leq n < \alpha$.
Pour $n = \alpha$, il découle de $PL^{n-1}(\mathcal{S}) \not\models l_j$ et $PL^n(\mathcal{S}) \models l_j$ qu'il existe une clause $l_j \vee l_j^1 \vee \cdots \vee l_j^p$ telle que $\mathcal{T}(\mathcal{P}_j) \models l_j \vee l_j^1 \vee \cdots \vee l_j^p$ et pour $i \in [1..p]$ $PL^n(\mathcal{S}) \models \bar{l}_j^i$. Par conqéquent, il découle facilement que $\mathcal{T}(\mathcal{P}_j) \cup \bigcup_{i=1}^p \{\bar{l}_j^i \mid PL^{n-1}(\mathcal{S}) \models l_i$ et $\mathbf{K}(l_i) \Rightarrow \bar{l}_j^i \in \mathcal{M}(\mathcal{S})\} \models l_j$, c.-à-d. $\Box \in$ DECA$_{\mathbf{K}}{}^j(\bar{l}_j, \emptyset, false)$. ■

6 Travaux connexes de la littérature

De nombreux travaux ont étudié le calcul de conséquences dans un cadre centralisé (*cf.* [20, 21, 23, 28, 7, 8] pour obtenir une liste exhaustive de références). En revanche, très peu de travaux se sont intéressés à ce problème dans un cadre distribué. Nous présentons ici ces quelques travaux.

[1, 2, 3] étudient le problème de calcul de conséquences de la Définition 4 dans des P2PIS propositionnels avec des mappings non orientés. Nous avons décrit ces travaux dans l'introduction et nous avons pointé dans la section 4 les similitudes et les différences avec le problème de calcul de conséquences dans des P2PIS avec des mappings orientés.

[4, 5, 6] développent une solution distribuée pour le calcul de conséquences dans une théorie clausale de la logique propositionnelle (ou de la logique du premier ordre). Cette théorie est tout d'abord partitionnée en un *graphe de partitions* dans lequel les sous-théories correspondant aux partitions sont connectées si elles partagent des variables. Une arête entre deux sous-théories est étiquetée avec les variables qu'elles partagent. Ces deux sous-théories peuvent alors échanger des formules en termes de ces variables. Le graphe de partitions est ensuite converti en un arbre en supprimant certaines arêtes, et en étendant les étiquettes d'autres arêtes afin de garantir que toute formule échangeable directement entre deux théories situées sur un cycle peut encore être échangée – éventuellement indirectement en parcourant ce qui reste du cycle – après la suppression du cycle. L'arbre ainsi obtenu est à la base d'un algorithme efficace de calcul de conséquences.
Il est important de remarquer que cette approche n'est pas utilisable dans le cas d'un P2PIS à cause de la phase de construction de l'arbre qui requiert que des acteurs du système aient une vision globale du P2PIS. En revanche, cette approche s'est avérée utile pour la mise en œuvre de systèmes à base de super pairs ayant cette vision globale du réseau (*cf.* [17]).

Enfin, [27, 26] proposent des P2PIS basés sur les logiques de description distribuées (*cf.* [9, 10]) dans lesquels les pairs sont constitués d'une TBox de la logique $\mathcal{SHIQ}$. Un pair peut établir des mappings entre deux concepts, l'un de sa TBox et un autre d'une TBox d'un autre pair, indiquant que le premier concept subsume (ou est subsumé par) le second. Une procédure distribuée pour le test de subsomption (c.-à-d. de test de conséquence) à base de tableaux est fournie pour des mappings entre des concepts atomiques et une TBox distribuée acyclique de $\mathcal{SHIQ}$.
Il est important de noter que la contrainte d'acyclicité de la TBox modélisée par le P2PIS est une sérieuse limitation à l'utilisation de cette méthode car aucun acteur d'un P2PIS n'est en mesure de la garantir.

7 Conclusion & perspectives

Dans cet article, nous avons posé les bases théoriques et algorithmiques pour le calcul de conséquences dans les P2PIS propositionnels avec mappings orientés. En particulier, nous avons défini le premier cadre logique pour modéliser ces P2PIS et nous avons proposé l'algorithme DECA$_{\mathbf{K}}$ de calcul de conséquences dans ces P2PIS. Ce travail est soutenu par France Télécom R&D via le Contrat de Recherche Externe MédiaD (Médiation Distribuée) dans lequel nous étudions des solutions pour déployer des services pair-à-pair d'échange et partage de connaissances.

La prochaine étape de ce travail consistera à effectuer une étude expérimentale rigoureuse et détaillée du passage à l'échelle de DECA$_{\mathbf{K}}$ dans l'esprit de [2], où nous avions étudié le passage à l'échelle de DECA pour le calcul de conséquences dans les P2PIS propositionnels avec mappings non orientés introduits dans [1].
Par la même occasion, nous exploiterons le lien sémantique établi dans la section 4 entre les P2PIS avec mappings orientés et avec mappings non orientés, afin de quantifier la différence de coût entre ces deux approches en termes du nombre et de la taille des communications entre pairs pour effectuer un calcul de conséquences. Ceci reviendra à comparer DECA et DECA$_{\mathbf{K}}$, selon ces critères, pour des mêmes clauses soumises aux mêmes pairs dans des P2PIS identiques modulo l'utilisation de l'opérateur $\mathbf{K}$ dans les mappings.

Une perspective plus théorique de ce travail est l'étude du calcul de conséquences dans des P2PIS avec mappings orientés, lorsque ces derniers sont inconsistants. En effet, nous avons vu dans la section 5 de cet article que les propriétés de DECA$_{\mathbf{K}}$ sont vérifiées dans un P2PIS insatisfiable, même si dans ce cas les résultats retournés sont inutiles : DECA$_{\mathbf{K}}$ retourne des impliqués clausaux, mais aucun n'est propre car il n'existe pas d'impliqués propres dans un P2PIS insatisfiable. Des résultats utiles dans de tels P2PIS seraient les impliqués clausaux premiers propres et *bien fondés* de la clause fournie en entrée,

c.-à-d. calculés à partir d'un sous-P2PIS satisfiable. Un point de départ pour ce futur travail est [13] qui étudie le calcul de conséquences dans les P2PIS insatisfiables de [1, 2, 3].

Références

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting. In *European Conference on Artificial Intelligence*, pages 945–946, 2004.

[2] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Scalability study of peer-to-peer consequence finding. In *International Joint Conference on Artificial Intelligence*, pages 351–356, 2005.

[3] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting : Application to the semantic web. *Journal of Artificial Intelligence Research*, 25 :269–314, 2006.

[4] E. Amir and S. McIlraith. Partition-based logical reasoning. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 389–400, 2000.

[5] E. Amir and S. McIlraith. Theorem proving with structured theories. In *International Joint Conference on Artificial Intelligence*, pages 625–631, 2001.

[6] E. Amir and S. McIlraith. Partition-based logical reasoning for first order and propositional theories. *Artificial Intelligence*, 162(1-2) :49–88, 2005.

[7] M. Bienvenu. Consequence finding in ALC. In *International Workshop on Description Logics*, pages 203–210, 2007.

[8] M. Bienvenu. Prime implicates and prime implicants in modal logic. In *AAAI*, pages 379–384, 2007.

[9] A. Borgida and L. Serafini. Distributed description logics : Directed domain correspondences in federated information sources. In *CoopIS/DOA/ODBASE*, pages 36–53, 2002.

[10] A. Borgida and L. Serafini. Distributed description logics : Assimilating information from peer sources. *Journal on Data Semantics*, 1 :153–184, 2003.

[11] C.-L. Chang and R.C.-T. Lee. *Symbolic Logic and mechanical Theorem Proving*. Academic Press, 1973.

[12] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical fondation of peer-to-peer data integration. In *Symposium on Principles Of Database Systems*, pages 241–251, 2004.

[13] P. Chatalic, G.-H. Nguyen, and M.-C. Rousset. Reasoning with inconsistencies in propositional peer-to-peer inference systems. In *European Conference on Artificial Intelligence*, pages 352–356, 2006.

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

[15] S. A. Cook. The complexity of theorem proving procedures. In *Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[16] E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. Queries and updates in the coDB peer-to-peer database system. In *International Conference on Very Large DataBases*, pages 1277–1280, 2004.

[17] François Goasdoué and Marie-Christine Rousset. Querying distributed data through distributed ontologies : A simple but scalable approach. *IEEE Intelligent Systems*, 18(5) :60–65, 2003.

[18] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(2) :319–379, 1992.

[19] G. Hughes and M. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[20] K. Inoue. Consequence-finding based on ordered linear resolution. In *International Joint Conference on Artificial Intelligence*, pages 158–164, 1991.

[21] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 2-3(56) :301–353, 1992.

[22] R. Ladner. Computational complexity of provability in systems of modal PL. *SIAM Journal on Computing*, 6(3) :467–480, 1977.

[23] P. Marquis. *Handbook on Defeasible Reasoning and Uncertainty Management Systems, chapter Consequence Finding*, volume 5, chapter Consequence Finding Algorithms, pages 41–145. Kluwer Academic Publisher, 2000.

[24] E. Minicozzi and R. Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence*, 3(1-3) :175–180, 1972.

[25] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1) :23–41, 1965.

[26] L. Serafini, A. Borgida, and A. Tamilin. Aspects of distributed and modular ontology reasoning. In *International Joint Conference on Artificial Intelligence*, pages 570–575, 2005.

[27] L. Serafini and A. Tamilin. DRAGO : Distributed reasoning architecture for the semantic web. In *European Semantic Web Conference*, pages 361–376, 2005.

[28] L. Simon and A. del Val. Efficient consequence finding. In *International Joint Conference on Artificial Intelligence*, pages 359–370, 2001.