Systèmes pair-à-pair sémantiques et extension non conservative d'une base de connaissances

Peer-to-Peer Semantic Systems and Non Conservative Extension of a Knowledge Base

Nada Abdallah et François Goasdoué
Univ. Paris-Sud & CNRS (LRI/IASI) – INRIA (Saclay-ÎdF/GEMO)
LRI, bâtiment 490, Univ. Paris-Sud, 91405 Orsay Cedex, France
{nada,fg}@lri.fr

Résumé

Cet article montre en quoi la notion d'extension non conservative d'une base de connaissances (KB) est importante dans les sytèmes d'inférence pair-à-pair (P2PIS), aussi connus sous le nom de systèmes pair-à-pair sémantiques. Cette notion est utile à un pair afin de détecter si (une partie de) sa KB est corrompue par un P2PIS ou pour apprendre du P2PIS de nouvelles connaissances sur son propre domaine d'application. Cette notion est d'autant plus importante qu'elle a des liens étroits avec la confidentialité des connaissances d'un pair au sein d'un P2PIS et avec la qualité de service fournie par un P2PIS. Nous étudions ici, du point de vue théorique et de l'algorithmique décentralisée, les deux problèmes suivants : (i) décider si un P2PIS est une extension conservative d'un pair donné et (ii) calculer les témoins d'une possible corruption de la KB d'un pair donné par un P2PIS, de sorte à pouvoir l'empêcher. Nous considérons des P2PIS passant à l'échelle d'un millier de pairs et dont l'utilité a déjà été démontrée en Intelligence Artificielle et en Bases de Données.

Mots-clefs : systèmes pair-à-pair sémantiques, extension (non) conservative, logique propositionnelle, déduction, algorithmique décentralisée.

Abstract

This paper points out that the notion of non conservative extension of a knowledge base (KB) is important to the distributed logical setting of peer-to-peer inference systems (P2PIS), a.k.a. peer-to-peer semantic systems. It is useful to a peer in order to detect/prevent that a P2PIS corrupts (part of) its knowledge or to learn more about its own application domain from the P2PIS. That notion is all the more important since it has connections with the privacy of a peer within a P2PIS and with the quality of service provided by a P2PIS. We therefore study the following tightly related problems from both the theoretical and decentralized algorithmic perspectives : (i) deciding whether a P2PIS is a conservative extension of a given peer and (ii) computing the witnesses to the corruption of a given peer's KB within a P2PIS so that we can forbid it. We consider here scalable P2PISs that have already proved useful to Artificial Intelligence and DataBases.

Keywords : peer-to-peer semantic systems, (non) conservative extension, propositional logic, deduction, decentralized algorithms.

1 Introduction

Ces dernières années, les *systèmes d'inférence pair-à-pair* (P2PIS), ou systèmes pair-à-pair sémantiques, ont reçu une attention considérable car leur infrastructure permet de raisonner sur des connaissances disséminées dans des réseaux. Notamment, ces systèmes ont été largement étudiés en *Intelligence Artificielle* (IA) et en *Bases de Données* (BD). Des P2PIS ont été étudiés en IA pour le calcul de conséquences dans des réseaux d'agents intelligents (par ex. [1, 2]) ou le test de subsomption dans des réseaux d'ontologies exprimées en logiques de description (par ex. [3, 4]). En BD, des P2PIS ont été étudiés pour répondre à des requêtes dans des réseaux de BD relationnelles (par ex. [5, 6, 7, 8, 9, 10, 11, 12]) ou dans le Web Sémantique (par ex. [13, 14, 1, 15]).
Les P2PIS sont des systèmes fondés sur la logique. Ils sont constitués de serveurs autonomes (c.-à-d. conçus et administrés indépendamment les uns des autres) appelés *pairs*. Chaque pair gère une base de connaissances (KB) modélisant son domaine d'application ou son champ d'expertise en termes de son *propre* vocabulaire (par ex. la théorie d'un agent, une base de données ou une ontologie). Des pairs ayant des centres d'intérêt similaires peuvent établir des correspondances sémantiques entre leurs KB en utilisant des formules logiques particulières appelées *mappings*. Ces mappings jouent un rôle essentiel puisque, d'une part, ils définissent comment les KB de certains pairs sont intégrées deux-à-deux et, d'autre part, ils donnent lieu à un réseau sémantique dans lequel il devient possible de raisonner (l'union des KB et des mappings de tous les pairs). En effet, une tâche d'inférence sur la KB d'un pair donné peut être effectuée sur la KB du P2PIS en se propageant de proche en proche dans le réseau de pairs en suivant les mappings appropriés.
Toutefois, raisonner dans un P2PIS n'est pas si simple. Le cadre pair-à-pair soulève des problèmes algorithmiques non triviaux puisqu'aucun pair d'un P2PIS n'a une vision globale du système : chaque pair ne connaît que sa propre KB et ses mappings avec d'autres pairs. Par conséquent, les algorithmes standards (c.-à-d. centralisés) de raisonnement *ne peuvent pas* être réutilisés dans les P2PIS : ils supposent que la KB sur laquelle la tâche de raisonnement doit être effectuée (ici la KB du P2PIS) est fournie en entrée. Le *challenge* est donc de définir des algorithmes décentralisés de raisonnement dont le but est de réduire une tâche d'inférence sur la KB du P2PIS à un calcul décentralisé entre pairs.

Cet article traite de la notion d'extension (non) conservative d'une KB dans le cadre des P2PIS. Formellement, étant données deux KB Σ_1 et Σ_2 utilisant la même logique, $\Sigma_1 \cup \Sigma_2$ est une extension conservative de Σ_1 si et seulement si toute conséquence logique de $\Sigma_1 \cup \Sigma_2$, utilisant *seulement* (un sous ensemble donné) des symboles de Σ_1, est aussi une conséquence de Σ_1. Ainsi, tester si l'extension de Σ_1 par Σ_2 est conservative exhibe si $\Sigma_1 \cup \Sigma_2$ fournit des connaissances que Σ_1 seule ne rend pas disponibles, alors que ces connaissances sont en termes de ses propres symboles (c.-à-d. dans son domaine de discours).
Le test d'extension conservative a été récemment le sujet d'un grand nombre de

travaux de recherche en logiques de description (par ex. [16, 17, 18, 19]) et en logiques modales (par ex. [20]), car il s'est révélé utile pour la maintenance, la réutilisation et l'intégration de KB. En particulier, il a été utilisé pour catégoriser certains types de mise-à-jour du cycle de vie d'une KB (par ex. [21]), comme le raffinement ou l'abstraction, et aussi pour définir la notion de modularité d'une KB (par ex. [22, 23, 24, 25]), l'équivalent de la notion de modularité en Ingénierie du Logiciel.

À notre connaissance, cet article est le premier à exhiber l'importance de la notion d'extension (non) conservative d'une KB dans le cadre logique distribué des P2PIS. En effet, un pair *étend* sa KB (à l'aide de mappings) avec celles des autres pairs (et leurs mappings) de sorte à pouvoir utiliser leurs connaissances pour répondre aux requêtes qui lui sont posées. Cette notion est d'autant plus importante ici qu'il est notoire que l'extension d'une KB n'est pas nécessairement conservative. Ainsi, lorsqu'un pair étend sa KB à l'aide de mappings, un P2PIS peut fournir des connaissances que ce pair *seul* ne fournit pas, alors que ces connaissances sont en termes de son *propre* vocabulaire, donc à propos de son *propre* domaine d'application ou champ d'expertise. Le problème est que de telles extra-connaissances sont disponibles au sein du P2PIS, que ce pair le veuille ou non. On peut distinguer les trois cas de figure suivants.

1. Supposons qu'un pair ait une description complète de son domaine d'application au sein de sa KB, par exemple un pair gérant une théorie axiomatisant les spécifications d'un composant dans une application pair-à-pair de diagnostic à base de modèles, une base de données dans une application pair-à-pair de e-commerce ou encore une ontologie "bien établie" dans une application pair-à-pair du Web Sémantique. Dans ce cas, la non conservativité de la KB de ce pair revient à une corruption de connaissances de la part du P2PIS. Ici, le pair devrait être – légitimement – en mesure d'interdire la corruption de ses connaissances.
2. Maintenant, supposons qu'un pair ait une description incomplète de son domaine d'application au sein de sa KB. Toutefois, ce pair est un expert sur (tout ou partie de) ce qui peut être dit en termes de son vocabulaire. Dans ce cas, la non conservativité de sa KB indique la présence d'extra-connaissances pour lesquelles ce pair est capable de décider si elles sont acceptables au sein du P2PIS. Ici, le pair devrait être – légitimement – en mesure d'interdire les extra-connaissances qu'il juge corruptrices au sein du système.
3. Enfin, considérons un pair ayant une description incomplète de son domaine d'application au sein de sa KB et aucune expertise particulière à propos de ce domaine. Dans ce cas, la non conservativité de sa KB indique simplement la présence de connaissances que ce pair peut être intéressé de connaître, afin d'enrichir ses connaissances sur son domaine d'application.

Il est important de remarquer que les 1. et 2. ci-dessus sont étroitement liés aux notions de *confidentialité* et de *qualité de service* (ou *QoS*). En effet, lorsqu'un

pair peut – de quelque façon que ce soit – interdire (tout ou partie) des extra-connaissances qu'un P2PIS a en termes de son vocabulaire, il a le contrôle de ce qui peut être dit ou non en termes de ce vocabulaire au sein du système. De plus, quand un pair expert sur son domaine d'application peut – d'une manière ou d'une autre – supprimer (tout ou partie) des extra-connaissances incorrectes qu'a le P2PIS en termes de son propre vocabulaire, la qualité des connaissances du P2PIS augmente, et donc la qualité des résultats de ses raisonnements (sur ces connaissances) augmente aussi.

Les motivations de cet article découlent de l'importance de la notion d'extension (non) conservative de la KB d'un pair au sein d'un P2PIS. Nous étudions les deux problèmes suivants du point de vue théorique et de l'algorithmique décentralisée : (i) décider si un P2PIS est une extension conservative d'un pair donné et (ii) calculer les témoins de la non conservativité d'un pair donné au sein d'un P2PIS, c'est-à-dire les conséquences logiques du P2PIS qui mettent en évidence une extention non conservative de la KB de ce pair. On notera que les témoins de non conservativité d'un pair caractérisent les extra-connaissances que le P2PIS a en termes du vocabulaire de ce pair, celles-ci pouvant être corruptrices pour ce pair. Toutefois, ces témoins résultent du fait que le pair a étendu sa KB avec des mappings. Ainsi, le pair peut supprimer les témoins non désirables en supprimant les mappings desquels ils résultent, en supposant bien évidemment qu'il sache de quels mappings ils résultent. Notre approche permet cela.

Plus précisément, notre contribution consiste à étudier les deux problèmes introduits ci-dessus *dans des P2PIS propositionnels*. Nous établissons leur *complexité* et nous les caractérisons à l'aide de techniques de calcul de conséquences afin de définir un *algorithme décentralisé* pour y répondre. Pour cela, nous proposons aussi le premier algorithme décentralisé de *déduction linéaire* dans un P2PIS propositionnel.

Étudier les deux problèmes énoncés ci-dessus dans le cadre de la *logique propositionnelle* (PL) poursuit l'idée que "small can be beautiful" [26] afin de fournir des services de raisonnement utiles, effectifs et passant à l'échelle de grands P2PIS. Notre crédo est que PL est *un* bon compromis entre expressivité et efficacité/passage à l'échelle des P2PIS.
Du point de vue de la calculabilité, l'utilisation de PL ne nécessite pas d'imposer des contraintes sur les P2PIS afin de conserver la décidabilité de raisonnements du cadre logique classique (c.-à-d. centralisé) [1]. En revanche, raisonner dans certains P2PIS utilisant des logiques plus expressives peut necéssiter de telles contraintes, celles-ci pouvant être perçues comme drastiques du point de vue d'applications réelles. Par exemple, il a été montré que répondre à une requête est indécidable dans un P2PIS relationnel, à moins de recourir à des contraintes topologiques (acyclicité des mappings) [5] ou encore à des sémantiques non standards (sémantiques épistémiques) [7, 8] avec lesquelles un mapping n'est plus une règle d'intégration entre deux pairs, mais une règle d'échange de connaissances entre un pair source et un pair cible.

Du point de vue de la complexité, les principales tâches de raisonnement en PL se situent au premier niveau de la hiérarchie polynomiale (par ex. [27, 28]). En revanche, d'autres logiques étudiées dans le cadre des P2PIS, et ne nécessitant pas de recourir à des contraintes sur les P2PIS afin de conserver la décidabilité de raisonnements du cadre logique classique (c.-à-d. centralisé), semblent trop expressives pour envisager des raisonnements efficaces et passant à l'échelle de grands P2PIS. Par exemple, la logique de description $\mathcal{SHIQ}$ utilisée dans les P2PIS de [3, 4] est ExpTime-complète pour SAT [29].
Enfin, du point de vues des applications réelles, les P2PIS propositionnels se sont déjà révélés utiles en IA et en BD. En ce qui concerne l'IA, le calcul (décentralisé) de conséquences a déjà été étudié dans ce cadre logique distribué [1]. Notons que l'inférence fondamentale de calcul de conséquences est utilisée en IA dans plusieurs tâches composites, telles que le raisonnement de sens commun, le diagnostic ou la compilation de connaissances. Ce calcul a été mis en œuvre dans la plateforme pair-à-pair SomeWhere et des expérimentations à partir de données synthétisées ont montré son passage à l'échelle jusqu'à des P2PIS d'un milier de pairs [30]. En ce qui concerne les BD, les P2PIS SomeOWL [1] et SomeRDFS [15] utilisent les P2PIS propositionnels mentionnés ci-dessus (via une étape de compilation de connaissances) pour répondre à une requête de façon décentralisée dans un réseau de pairs annotant des données avec leurs propres ontologies. Ces sytèmes sont construits comme une sur-couche de SomeWhere et leur modèle de données respectif est un fragment (propositionnel) d'une recommandation du W3C pour le Web Sémantique : OWL-PL pour SomeOWL, c.-à-d. le fragment $\mathcal{CLU}$ de la logique de description OWL-DL (la recommandation du W3C pour les ontologies complexes), et coreRDFS pour SomeRDFS, c.-à-d. le fragment de logique de description avec concepts et rôles[1,2] de RDFS (la recommandation du W3C pour les ontologies simples).

L'article est organisé comme suit. Nous définissons les P2PIS que nous considérons dans la Section 2. Nous spécifions les problèmes à étudier et établissons leur complexité dans la Section 3. Dans la Section 4, nous utilisons des techniques de calcul de conséquences de sorte à exhiber une manière effective et efficace de résoudre nos deux problèmes. Dans la Section 5, nous fournissons l'algorithme décentralisé CECA qui répond à ces problèmes. Enfin, nous concluons avec la présentation de travaux connexes de la littérature dans la Section 6 et avec la suite à donner à nos travaux dans la Section 7.

2 P2PIS

Un P2PIS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ est un ensemble de pairs, où l'indice i modélise l'*identifiant* du pair $\mathcal{P}_i$ dans $\mathcal{S}$ (par ex. son adresse IP).

[1]En logiques de description, les concepts sont des relations unaires et les rôles sont des relations binaires [31].

[2]coreRDFS correspond aussi à un fragment de la logique de description DL-lite$_\mathcal{R}$. DL-lite$_\mathcal{R}$ appartient à la famille des logiques DL-lite qui a été définies pour permettre des raisonnements efficaces sur des données annotées par des ontologies [32].

Un pair $\mathcal{P}_i$ gère des connaissances modélisées par une *théorie* propositionnelle, $\mathcal{T}(\mathcal{P}_i)$, et un *ensemble de mappings* avec d'autres pairs, $\mathcal{M}(\mathcal{P}_i)$.
La théorie de $\mathcal{P}_i$ est un sous-ensemble de son *langage*, $\mathcal{L}(\mathcal{P}_i)$. Ce langage est l'ensemble fini de clauses sans répétition de littéral exprimées en termes de l'*alphabet* propre de $\mathcal{P}_i$, $\mathcal{A}(\mathcal{P}_i)$. Un tel alphabet est fait de *variables* propositionnelles. Les alphabets des pairs étant disjoints deux-à-deux, nous indiçons chaque variable d'un pair par l'identifiant de celui-ci (par ex. la variable A du pair $\mathcal{P}_i$ est notée A_i). Nous distinguons un sous-ensemble $Tgt(\mathcal{A}(\mathcal{P}_i))$ de $\mathcal{A}(\mathcal{P}_i)$ qui représente des *variables cibles*. Ces variables définissent le *langage cible* de $\mathcal{P}_i$, $Tgt(\mathcal{L}(\mathcal{P}_i))$, qui est l'ensemble fini de clauses sans répétition de littéral exprimées en termes de $Tgt(\mathcal{A}(\mathcal{P}_i))$. Un langage cible donné est utilisé pour définir quelle partie du langage d'un pair est pertinente pour une application donnée, ici pour tester l'extension conservative seulement par rapport à un sous-langage donné : le sous-langage pour lequel $\mathcal{P}_i$ veut vérifier la corruption de ses connaissances, (in)valider les connaissances du P2PIS ou apprendre de nouvelles connaissances du P2PIS. Nous supposons que la clause vide $\square$ appartient à la fois à $\mathcal{L}(\mathcal{P}_i)$ et à $Tgt(\mathcal{L}(\mathcal{P}_i))$.
Les mappings dans lesquels $\mathcal{P}_i$ est impliqué sont stockés localement dans $\mathcal{M}(\mathcal{P}_i)$, c'est-à-dire que tout mapping entre deux pairs est stocké par ces deux pairs. Tout mapping entre les pairs $\mathcal{P}_i$ et $\mathcal{P}_j$ est de la forme $l_i \vee l_j$, tel que $l_i \in \mathcal{L}(\mathcal{P}_i)$ et $l_j \in \mathcal{L}(\mathcal{P}_j)$ sont des littéraux et $i \neq j$. Un tel mapping définit quatre littéraux *partagés* par $\mathcal{P}_i$ et $\mathcal{P}_j$: l_i, l_j, et leurs complémentaires $\bar{l}_i$ et $\bar{l}_j$.

La *théorie* d'un P2PIS $\mathcal{S}$, notée $\mathcal{T}(\mathcal{S})$, est l'union des théories de ses pairs. L'ensemble des *mappings* de $\mathcal{S}$, noté $\mathcal{M}(\mathcal{S})$, est l'union des mappings de ses pairs. Notons que d'un point de vue logique, un P2PIS $\mathcal{S}$ est la théorie propositionnelle clausale $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$.

La sémantique d'un pair $\mathcal{P}_i$ et d'un P2PIS $\mathcal{S}$ découle – bien évidemment – de celle des théories $\mathcal{T}(\mathcal{P}_i)$ et $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$.

Exemple La Figure 1 illustre le P2PIS $\mathcal{S}$ constitué des pairs $\mathcal{P}_1$, $\mathcal{P}_2$ et $\mathcal{P}_3$. Les théories des pairs sont les noeuds étiquetés avec les noms des pairs et les mappings entre deux pairs étiquettent les arêtes qui lient leur théorie respective. Nous supposons ici que *toutes* les variables sont cibles.

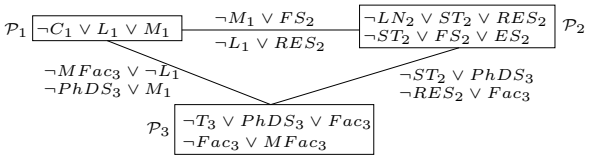

FIG. 1 – Le P2PIS $\mathcal{S}$.

Le pair $\mathcal{P}_1$ décrit un service de cours privés (C) offrant des cours de langues (L) et des cours de mathématiques (M). Le pair $\mathcal{P}_2$ gère des connaissances à propos d'apprentis (LN) qui sont des étudiants (ST) ou des chercheurs (RES). De plus, les étudiants sont francophones (FS) ou anglophones (ES). Le pair $\mathcal{P}_3$ stocke des connaissances à propos d'enseignants (T) qui sont soit des doctorants ($PhDS$), soit des (enseignants) permanents (Fac), tous les permanents étant des mathématiciens ($MFac$).

Les pairs $\mathcal{P}_1$ et $\mathcal{P}_2$ partagent le fait que les cours de mathématiques sont pour les francophones et que les cours de langues sont pour les chercheurs. Le pair $\mathcal{P}_2$ partage

aussi avec le pair $\mathcal{P}_3$ le fait que les étudiants préfèrent avoir des doctorants comme enseignants et les chercheurs préfèrent avoir des permanents. Enfin, les pairs $\mathcal{P}_1$ et $\mathcal{P}_3$ partagent le fait que les permanents mathématiciens ne donnent pas de cours de langues et que les doctorants enseignent les mathématiques. ■

3 Problèmes

Nous étudions ici les problèmes consistant, d'une part, à *décider si un P2PIS est une extension conservative d'un pair donné* (problème de décision) et, d'autre part, à *calculer les conséquences d'un P2PIS exhibant la non conservativité d'un pair donné* (problème fonctionnel). Tout d'abord, nous définissons formellement la notion d'*extension conservative d'un pair au sein un P2PIS*, celle-ci étant fondée sur la notion de *conséquence* logique. Nous posons ensuite notre problème de décision, puis nous établissons sa complexité. Enfin, nous posons notre problème fonctionnel.

Définition 1 (Conséquence) *Étant donnés un alphabet $\mathcal{A}$ de variables propositionnelles, une théorie propositionnelle $\mathcal{T}$ de formules en termes de $\mathcal{A}$ et deux formules propositionnelles f et g en termes de $\mathcal{A}$:*

- *f est une* conséquence *de g, $g \models f$, ssi tout modèle de g est aussi un modèle de f et*
- *f est une* conséquence *de $\mathcal{T}$, $\mathcal{T} \models f$, ssi tout modèle de $\mathcal{T}$ est aussi un modèle de f.*

Définition 2 (Extension conservative) *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$. $\mathcal{S}$ est une* extension conservative *de $\mathcal{P}$ ssi la propriété Ψ est vérifiée.*
Ψ : pour tout $c \in Tgt(\mathcal{L}(\mathcal{P}))$, si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ alors $\mathcal{T}(\mathcal{P}) \models c$.

Ceux qui sont familiers avec la notion d'*interpolation uniforme*, une variante de l'interpolation de Carig [33], pourront remarquer le lien étroit existant avec la notion d'extension conservative (par ex. [34]). En effet, dans le cadre de nos P2PIS, $\mathcal{S}$ est une extension conservative de $\mathcal{P}$ ssi $\mathcal{T}(\mathcal{P})$ est un interpolant uniforme de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ par rapport à $Tgt(\mathcal{L}(\mathcal{P}))$.

Partant de la définition ci-dessus, notre problème de décision est le suivant.

Définition 3 (Problème de décision)
DONNÉES : *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$.*
QUESTION : *Est-ce que $\mathcal{S}$ est une extension conservative de $\mathcal{P}$?*

On peut voir – à partir de la Définition 2 – que ce problème est difficile. Il consiste en effet à résoudre des problèmes coNP-complets[3] pour un nombre exponentiel de clauses[4] (dans le pire des cas). Cette intuition est confirmée par le théorème suivant qui établit la complexité exacte de notre problème de décision.

[3] Décider si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$, resp. $\mathcal{T}(\mathcal{P}) \models c$, revient à décider si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{\neg c\} \models \square$, resp. $\mathcal{T}(\mathcal{P}) \cup \{\neg c\} \models \square$, c.-à-d. à répondre à UNSAT.

[4] En fait, $3^{cardinal(\mathcal{A}(\mathcal{P}))}$ clauses dans le pire des cas puisque nous pouvons générer toute clause de $\mathcal{L}(\mathcal{P})$, donc de $Tgt(\mathcal{L}(\mathcal{P}))$, en combinant de façon disjonctive soit V, $\neg V$ ou $\square$ pour chaque variable V de $\mathcal{A}(\mathcal{P})$.

Théorème 1 (Complexité) *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$. Décider si $\mathcal{S}$ est une extension conservative de $\mathcal{P}$ est Π_2^p-complet.*

Ce problème est Π_2^p-facile car le problème complémentaire est dans Σ_2^p : il suffit de deviner une clause sans répétition de littéral et exprimée en termes de l'alphabet cible de $\mathcal{P}$, puis de vérifier en appelant un NP-oracle que $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ et $\mathcal{T}(\mathcal{P}) \not\models c$. Ce problème est Π_2^p-difficile du fait d'une réduction polynomiale depuis le problème de validité d'une formule $QBF_{2,\forall}$, le problème canonique de la classe Π_2^p. Une formule $QBF_{2,\forall}$ est de la forme $\forall A \exists B \Phi$ où A et B forment une partition des variables de la formule propositionnelle Φ. Puisque $\forall A \exists B \Phi$ est valide si et seulement si il n'existe pas de conséquence logique de Φ exprimée uniquement en termes de variables de A [28], une preuve de Π_2^p-difficulté pour notre problème revient à construire un P2PIS tel que $\forall A \exists B \Phi$ est valide si et seulement si le P2PIS est une extension conservative d'un pair donné lorsqu'il n'existe pas de conséquence de Φ exprimée en termes de A. Par exemple, puisque Φ peut être vu, sans perte de généralité, comme la théorie propositionnelle clausale qui lui est équivalente, considérons le P2PIS suivant constitué des deux pairs $\mathcal{P}_1$ et $\mathcal{P}_2$: $\mathcal{T}(\mathcal{P}_1) = \{new\}$, $\mathcal{T}(\mathcal{P}_2) = \Phi$, $\mathcal{M}(\mathcal{P}_1) = \mathcal{M}(\mathcal{P}_2) = \bigcup_{i=1}^{n} \{\neg A_1^i \vee A_2^i, \neg A_2^i \vee A_1^i\}$ avec $Tgt(\mathcal{A}(\mathcal{P}_1)) = \{A_1^1, \ldots, A_1^n\}$, $\mathcal{A}(\mathcal{P}_1) = Tgt(\mathcal{A}(\mathcal{P}_1)) \cup \{new\}$, $\mathcal{A}(\mathcal{P}_2) = Tgt(\mathcal{A}(\mathcal{P}_2)) = A \cup B$ et $A = \{A_2^1, \ldots, A_2^n\}$. Il découle alors que $\forall A \exists B \Phi$ est valide (c.-à-d. il n'y a pas de conséquence de Φ exprimée uniquement en termes de A) si et seulement si ce P2PIS est une extension conservative de $\mathcal{P}_1$.

Notre problème fonctionnel peut être vu comme une façon d'expliquer la non conservativité d'un pair. Il consiste à énumérer tous les *témoins* de non conservativité d'un pair, c'est-à-dire tous les contre-exemples à la conservativié de ce pair. Remarquons qu'en résolvant notre problème fonctionnel, nous résolvons aussi notre problème de décision. En effet, pour un pair donné, un ensemble vide de témoins montre sa conservativité et un ensemble non vide de témoins montre sa non conservativité.

Définition 4 (Témoin) *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$. Soit c une clause de $Tgt(\mathcal{L}(\mathcal{P}))$. c est un* témoin *de l'extension non conservative $\mathcal{P}$ au sein de $\mathcal{S}$ ssi $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ et $\mathcal{T}(\mathcal{P}) \not\models c$.*

Définition 5 (Problème fonctionnel)
DONNÉES : *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$.*
QUESTION : *Quels sont les témoins de l'extension non conservative de $\mathcal{P}$ au sein de $\mathcal{S}$?*

Exemple (suite) Le P2PIS de la Figure 1 n'est ni une extenstion conservative de $\mathcal{P}_1$, ni une extenstion conservative de $\mathcal{P}_2$. (Il est toutefois une extension conservative de $\mathcal{P}_3$.) En effet, on peut vérifier (par ex. par réfutation) que $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \neg L_1$ alors que $\mathcal{T}(\mathcal{P}_1) \not\models \neg L_1$, et que $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models \neg ST_2 \vee FS_2$ alors que $\mathcal{T}(\mathcal{P}_2) \not\models \neg ST_2 \vee FS_2$. Notons que $\neg L_1$ est *incomparable* avec les connaissances de $\mathcal{T}(\mathcal{P}_1)$, alors que $\neg ST_2 \vee FS_2$ est *subsumé par* une clause de $\mathcal{T}(\mathcal{P}_2)$. Ceci impose donc à $\mathcal{P}_1$ de ne proposer que des cours de mathématiques et à $\mathcal{P}_2$ de n'avoir que des étudiants francophones ! ■

4 Techniques

Pour résoudre notre problème de décision et notre problème fonctionnel, nous allons tout d'abord calculer un ensemble de conséquences du P2PIS dans le langage cible d'un pair donné, puis tester si ces conséquences sont aussi des conséquences de ce pair seul (problème de décision) ou bien quelles conséquences sont des témoins de la non conservativité de ce pair (problème fonctionnel). Évidemment, nous garantissons que l'ensemble des conséquences calculées est toujours suffisant pour répondre correctement à nos deux problèmes. La question qui se pose est alors *comment* calculer de façon effective – et si possible efficace – un tel ensemble de conséquences dans notre cadre logique *distribué*.

Nous recourons aux techniques (décentralisées) de calcul de conséquences afin de fournir une caractérisation *exacte* des conséquences à calculer, et *comment* celles-ci peuvent être calculées. Pour cela, nous réutilisons la notion bien connue d'*impliqués premiers d'une théorie propositionnelle* (c.-à-d. ses conséquences les plus fortes).

Définition 6 (Impliqué premier) *Soit $\mathcal{T}$ une théorie propositionnelle consituée de formules en termes d'un alphabet $\mathcal{A}$, et soit f une clause exprimée en termes de $\mathcal{A}$.*

- *f est un* impliqué de $\mathcal{T}$ *ssi* $\mathcal{T} \models f$.
- *f est un* impliqué premier de $\mathcal{T}$ *ssi f est un impliqué de $\mathcal{T}$ et pour toute autre clause f' exprimée en termes de $\mathcal{A}$, si $\mathcal{T} \models f'$ et $f' \models f$ alors $f \models f'$ (c.-à-d. $f' \equiv f$).*

Notons que la résolution [35] est un mécanisme clef pour trouver des impliqués (premiers) d'une théorie propositionnelle clausale. En particulier, les stratégies de résolution sont *correctes* pour le calcul de conséquences dans une telle théorie, plusieurs d'entres elles étant aussi *complètes*. Une stratégie de résolution est correcte pour le calcul de conséquences si elle ne dérive que des impliqués d'une théorie. Elle est complète si elle peut dériver tout impliqué *premier* d'une théorie. Dans la suite, nous considérons la résolution et la résolution linéraire [36] qui sont correctes et complètes pour le calcul de conséquences [37, 38, 39, 28]. On notera par $\mathcal{T} \vdash_R c$ le fait qu'il existe une *déduction* (ou dérivation) de c – fondée sur la résolution – dans la théorie propositionnelle clausale $\mathcal{T}$ et par $\mathcal{T} \vdash^r_{LR} c$ le fait qu'il existe une *déduction linéaire*[5] (ou dérivation linéaire) de c – fondée sur la résolution linéraire – dans la théorie propositionnelle clausale $\mathcal{T}$ et dont la *top clause* est $r \in \mathcal{T}$.

Il est notoire que toute théorie propositionnelle (clausale) est équivalente à l'ensemble de ses impliqués premiers[6] [28]. Par conséquent, il découle de la Définition 6 que trouver l'ensemble des impliqués premiers d'un P2PIS exprimés dans le langage cible d'un pair donné est *suffisant* pour répondre à nos deux problèmes. Toutefois, le nombre d'impliqués premiers d'une théorie (clausale) peut être exponentiel par rapport à la taille de cette théorie (par ex. [28]) ! Fort heureusement, d'un point de vue "optimi-

[5] Dans la suite, nous utilisons la terminologie de [36] pour parler de la déduction linéraire (par ex. top clause, center clause ou encore side clause).

[6] Nous notons $\mathbf{PI}(\mathcal{T})$ l'ensemble des impliqués premiers de la théorie propositionnelle $\mathcal{T}$.

sation", il s'avère que trouver *tous* ces impliqués *n'est pas nécessaire*. La Proposition 1 établit en effet que nous avons *seulement* à considérer les impliqués premiers découlant *nécessairement* des mappings de ce pair. La preuve consiste à montrer que tout autre impliqué premier est sans intérêt pour le test d'extension conservative : soit c'est un impliqué de ce pair seul (c.-à-d. il satisfait alors nécessairement la propriété Ψ de la Définition 3), soit il n'est pas exprimé dans le language (cible) de ce pair (c.-à-d. il n'a alors rien à voir avec la propriété Ψ de la Définition 3).

Proposition 1 *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}$. Si $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}))$ et $\mathcal{T}(\mathcal{P}) \not\models c$ avec $c \in Tgt(\mathcal{L}(\mathcal{P}))$ alors* toute *déduction linéaire de c utilise un mapping de $\mathcal{M}(\mathcal{P})$.*

La Proposition 2 fournit une manière effective de calculer les impliqués premiers nécessaires pour tester si un P2PIS est une extension conservative d'un pair donné. Elle montre que ces impliqués premiers peuvent être calculés à l'aide de déductions linéraires dont les top clauses sont les mappings de ce pair. Puisque toute déduction linéaire d'un tel impliqué utilise un mapping de ce pair (Proposition 1), la preuve consiste à montrer que lorsqu'un mapping est utilisé comme side clause dans une déduction linéraire d'un impliqué, il existe une autre déduction linéaire de cet impliqué dans laquelle ce mapping est la top clause.

Proposition 2 *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}$. Si $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}))$ et $\mathcal{T}(\mathcal{P}) \not\models c$ avec $c \in Tgt(\mathcal{L}(\mathcal{P}))$ alors il existe $m \in \mathcal{M}(\mathcal{P})$ tel que $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^m_{LR} c$.*

Remarquons que la Proposition 1 et la Proposition 2 sont vraies *uniquement* pour des P2PIS *satisfiables*. La raison à cela est qu'un P2PIS insatisfiable a un unique impliqué premier, la clause vide $\square$, qui appartient au langage cible de tous les pairs. Par conséquent, ces propositions seraient vérifiées si $\square$ pouvait être dérivée (par déduction linéaire) à partir d'un mapping de chaque pair, ce qui n'est – bien évidemment – pas le cas (par ex. pour un pair qui n'est pas impliqué dans l'insatisfiabilité). Toutefois, ceci ne pose pas un réel problème. En effet, nous élaborerons, dans la section suivante, une stratégie (décentralisée) auto-guérissante permettant de rendre satisfiable un P2PIS insatisfiable.

À partir des propriétés de la déduction (linéaire) et de la Proposition 2, nous aboutissons finalement au théorème suivant qui reformule la notion d'extension conservative d'un pair en termes de techniques de calcul de conséquences.

Théorème 2 (Reformulation) *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}$. $\mathcal{S}$ est une* extension conservative *de $\mathcal{P}$ ssi la propriété suivante Ψ' est vérifiée.*
Ψ' : pour tout mapping $m \in \mathcal{M}(\mathcal{P})$, si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^m_{LR} c$ et $c \in Tgt(\mathcal{L}(\mathcal{P}))$ alors $\mathcal{T}(\mathcal{P}) \cup \{\neg c\} \vdash_R \square$.

Il découle du théorème ci-dessus que résoudre nos deux problèmes pour un pair donné nécessite un *algorithme décentralisé de déduction linéaire* (pour calculer $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^m_{LR} c$) et un *algorithme centralisé de déduction* (pour tester $\mathcal{T}(\mathcal{P}) \cup \{\neg c\} \vdash_R \square$). On notera qu'il existe déjà plusieurs solutions pour la déduction/réfutation dans

une théorie de PL (par ex. [36, 28]), tandis que la littérature ne fournit aucune solution pour la déduction linéaire dans un P2PIS.

5 Algorithmes

Nous proposons ici deux algorithmes. Un algorithme décentralisé de calcul de conséquences fondé sur la résolution linéaire (DeCA$_{LR}$, Algorithme 1) et un algorithme de test d'extension conservative d'un pair (CECA, Algorithme 2) qui recoure à DeCA$_{LR}$ en suivant le théorème 2. Notons que le deuxième algorithme est capable de résoudre à la fois notre problème de décision et notre problème fonctionnel.

Déduction linéaire décentralisée. Deux algorithmes décentralisés de déduction dans des P2PISs propositionnels ont déjà été proposés dans la littérature.
Le premier algorithme a été conçu afin d'*optimiser* le calcul de conséquences dans *une* théorie de PL (ou de la logique du premier ordre) [40, 41, 42]. Toutefois, il pourrait aussi être utilisé dans *certains* P2PIS. L'optimisation proposée consiste à partitionner une théorie en un P2PIS propositionnel ayant une *topologie d'arbre* avant d'appliquer un algorithme décentralisé de déduction, appelé Message-Passing (MP), recourant à cette topologie. Ce qui est intéressant dans cette approche est que MP n'exige pas qu'un agent ait une vue globale du système à l'inverse de l'étape de partitionnement. Il peut alors être utilisé pour la déduction dans des P2PIS. Toutefois, la limitation sévère à son utilisation dans des applications réelles est que les P2PIS doivent avoir une topologie d'arbre.
L'autre algorithme, appelé Decentralized Consequence finding Algorithm (DeCA), est le premier algorithme spécialement conçu pour la déduction dans des P2PISs propositionnels, sans avoir à imposer de contrainte sur la topologie du réseau de pairs [1].

DeCA$_{LR}$ (Decentralized Consequence finding Algorithm based on Linear Resolution, Algorithme 1) est à DeCA ce que la déduction linéaire est à la deduction. C'est le *premier* algorithme décentralisé pour la déduction *linéaire* dans un P2PIS propositionnel et il ne nécessite pas d'imposer des contraintes sur la topologie du réseau de pairs. DeCA$_{LR}$ calcule des impliqués d'un P2PIS, dont *tous* les impliqués premiers pouvant être dérivés par *déductions linéaires* (décentralisées) à partir d'une top clause exprimée dans le langage d'*un* pair.

DeCA$_{LR}$ est un algorithme récursif décentralisé qui s'exécute sur chacun des pairs d'un P2PIS, la signification d'un appel récursif étant qu'un pair en sollicite un autre pour calculer des conséquences d'une clause donnée. Le point subtile dans l'Algorithme 1 est que l'instance de DeCA$_{LR}$ s'exécutant sur le pair $\mathcal{P}_k$ est notée DeCA$_{LR}{}^k$. Ainsi, dans DeCA$_{LR}$, un pair $\mathcal{P}_k$ sollicite un autre pair $\mathcal{P}_j$ par un appel récursif à DeCA$_{LR}{}^j$. Nous adoptons cette notation dans un souci de clarté et afin de nous abstraire de tout choix d'implémentation particulier (par ex. passage de messages par sockets, Services Web ou middlewares distribués t.q. JAVA-RMI ou CORBA).

Un élément clef de DeCA$_{LR}$ est l'utilisation d'un historique (*hist*) qui garde la

trace des clauses pour lesquelles les pairs sont sollicités au sein d'appels récursifs imbriqués (dans l'esprit d'une pile d'appels). L'historique fournit un contexte de raisonnement pour un pair : celui-ci est sollicité pour calculer des conséquences d'une clause donnée car les clauses de l'historique sont vraies dans le système. Un tel historique est utile dans notre cadre distribué car il peut arriver qu'un même pair soit sollicité plusieurs fois au sein d'appels récursifs imbriqués. Un tel pair doit alors se souvenir à chaque sollicitation des clauses pour lesquelles il a déjà été sollicité dans les appels récursifs précédents : elles sont vraies dans le système et donc ce pair doit les utiliser pour raisonner. De plus, cet historique est aussi utilisé pour détecter et empêcher les récursions infinies qui se produisent lorsqu'un même pair est sollicité plus d'une fois pour une *même* clause au sein d'appels récursifs imbriqués.

Nous donnons maintenant un aperçu du calcul effectué DeCA$_{LR}$. Étant donnée une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, DeCA$_{LR}{}^k$ (DeCA$_{LR}$ sur $\mathcal{P}_k$) calcule tout impliqué (premier) pouvant être obtenu de $\mathcal{T}(\mathcal{P}_k) \cup hist \cup \{q_k\}$ par une déduction linéaire dont la top clause est q_k (ligne 4). Dans un second temps (ligne 9), DeCA$_{LR}{}^k$ calcule de nouveaux impliqués (premiers) à partir de ceux obtenus dans la première étape, en utilisant l'opérateur de distribution clausale $\otimes$[7]. DeCA$_{LR}{}^k$ commence par utiliser les mappings pour solliciter des pairs – exécutant le même algorithme – afin de calculer des impliqués (premiers) des littéraux partagés d'un impliqué issu de la première étape. DeCA$_{LR}{}^k$ construit alors de nouveaux impliqués (premiers) en combinant par $\otimes$ les littéraux non partagés de l'impliqué issu de la première étape (s'il y en a) et chaque ensemble d'impliqués (premiers) calculé à partir de ses littéraux partagés (ligne 10).
Évidemment, DeCA$_{LR}{}^k$ utilise les conditions d'arrêt appropriées pour garantir la terminaison en présence de raisonnement cyclique (ligne 1) ou pour éviter des calculs inutiles (lignes 5 et 11). DeCA$_{LR}{}^k$ utilise aussi les conditions de filtrage appropriées afin de ne produire que des clauses dans le langage d'un pair donné $\mathcal{P}_{id}$ du P2PIS (lignes 8 et 13), lorsque DeCA$_{LR}$ est utilisé par CECA (Algorithme 2).

Algorithme 1: DeCA$_{LR}$ sur le pair $\mathcal{P}_k$ du P2PIS $\mathcal{S}$. [Le texte entre crochets ajoute des *optimisations* lorsque DeCA$_{LR}$ est appelé par l'algorithme CECA (Algorithme 2).]

DeCA$_{LR}{}^k(q_k, hist[, id])$
Entrée: une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, un ensemble de clauses $hist$ [et un identifiant de pair id]
Sortie: un ensemble d'impliqués (premiers) de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \cup hist$ [appartenant à $\mathcal{L}(\mathcal{P}_{id})$]
(1) **if** $q_k \in hist$ (c.-à-d. q_k a déjà été traité)
(2) **return** $\emptyset$ // plus de nouvel impliqué (cycle)
(3) **else**
(4) $CI \leftarrow \{q_k\} \cup \{c \mid \mathcal{T}(\mathcal{P}_k) \cup hist \cup \{q_k\} \vdash_{LR}^{q_k} c\}$
(5) **if** $\square \in CI$
(6) **return** $\{\square\}$ // plus de nouvel impliqué, seul $\square$ est premier
(7) **else**
(8) [**if** $k \neq id$ retirer de CI toute clause ayant un littéral $l \in \mathcal{L}(\mathcal{P}_k)$ non partagé]
(9) **foreach** clause $c = l_k^1 \vee \cdots \vee l_k^q \vee l_k^{q+1} \vee \cdots \vee l_k^n \in CI$ t.q. [si $k \neq id$ alors $l_k^1 \vee \cdots \vee l_k^q = \square$, sinon] soit $l_k^1 \vee \cdots \vee l_k^q$ est fait de littéraux non partagés de c s'il y en a, ou soit $l_k^1 \vee \cdots \vee l_k^q = \square$
(10) $CI = CI \cup \{l_k^1 \vee \cdots \vee l_k^q\} \otimes \otimes_{i=q+1}^{n} [\{l_k^i\} \cup \{c \mid c \in$ DeCA$_{LR}{}^j(l_j, hist \cup \{q_k\}[, id])$ and $\overline{l_k^i} \vee l_j \in \mathcal{M}(\mathcal{P}_k)\}]$
(11) **if** $\square \in CI$
(12) **return** $\{\square\}$ // seul $\square$ est premier
(13) [retirer de CI toute clause n'appartenant pas à $\mathcal{L}(\mathcal{P}_{id})$]
(14) **return** CI

[7] L'opérateur de distribution clausale $\otimes$ est associatif et commutatif. Il s'applique à deux ensembles de clauses et produit un ensemble de clauses *sans* répétition de littéral (il les supprime) tel que $\emptyset \otimes \mathcal{C} = \emptyset$, $\{\square\} \otimes \mathcal{C} = \mathcal{C}$ et $\{c^1, \ldots, c^m\} \otimes \{c_1, \ldots, c_n\} = \{c^1 \vee c_1, \ldots, c^1 \vee c_n, \ldots, c^m \vee c_1, \ldots, c^m \vee c_n\}$.

Le Théorème 3 énonce les propriétés de DeCA$_{LR}$. Le texte entre crochets n'est à considérer que si on considère aussi le texte entre crochets dans DeCA$_{LR}$. Ce théorème établit que DeCA$_{LR}$ est une procédure effective pour la déduction linéaire décentralisée [dans le langage du pair interrogé], dont la top clause est dans le langage du pair interrogé.

Théorème 3 *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}_k$. Soit la clause q_k appartenant à $\mathcal{L}(\mathcal{P}_k)$.*

- DeCA$_{LR}{}^k(q_k, \emptyset[, k])$ *s'arrête.*
- *Si* $c \in$ DeCA$_{LR}{}^k(q_k, \emptyset[, k])$ *alors* $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ *[et* $c \in \mathcal{L}(\mathcal{P}_k)$*].*
- *Si* $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$*,* $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}))$ *[et* $c \in \mathcal{L}(\mathcal{P}_k)$*] alors* $c \in$ DeCA$_{LR}{}^k(q_k, \emptyset[, k])$*.*

La terminaison découle du fait que si DeCA$_{LR}$ ne s'arrête pas, il existe un nombre infini d'appels récursifs et donc un historique infini. Ceci est impossible puisque DeCA$_{LR}$ s'arrête lorsque la même clause est traitée deux fois au sein d'appels récursifs imbriqués (ligne 2) et le nombre de clauses possibles à traiter est fini puisque le nombre de mappings est fini. Concernant la correction, elle est prouvée par récurrence sur le nombre d'appels récursifs à DeCA$_{LR}$ nécessaires pour dériver c. Enfin, la complétude est prouvée par récurrence sur le nombre n de mappings apparaissant dans un ensemble minimal de clauses d'une déduction linéaire de c (c.-à-d. dans lequel supprimer une clause ne permet plus de prouver c) : DeCA$_{LR}$ produit c en utilisant n appels récursifs. Notons que la correction et la complétude s'appuient sur la Proposition 3 (pour la ligne 10). Cette dernière découle de la correction et de la complétude du calcul de conséquences par déduction linéaire, ainsi que de la propriété de *distribution* offrant une caractérisation des impliqués premiers (Proposition 84 dans [28]).

Proposition 3 *Soit $\mathcal{T}$ une théorie propositionnelle clausale dont une clause est $l_1 \vee \cdots \vee l_n$.*

- *Si* $c \in \Diamond_{i=1}^{n}\{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$ *alors* $\mathcal{T} \vdash_{LR}^{l_1 \vee \cdots \vee l_n} c$*.*
- *Si* $\mathcal{T} \vdash_{LR}^{l_1 \vee \cdots \vee l_n} c$ *et* $c \in \mathbf{PI}(\mathcal{T})$ *alors* $c \in \Diamond_{i=1}^{n}\{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$*.*

Extension conservative décentralisée. Maintenant que nous avons un algorithme décentralisé de calcul de conséquences fondé sur la résolution linéaire, nous proposons CECA (Algorithme 2), pour Conservative Extension Checking Algorithm, afin de résoudre nos deux problèmes.

En suivant le théorème 2, CECA^k (CECA sur $\mathcal{P}_k$) considère chaque mapping de $\mathcal{M}(\mathcal{P}_k)$ (ligne 2) à partir duquel il calcule des impliqués (premiers) du P2PIS en utilisant DeCA$_{LR}$ comme outil de déduction linéaire décentralisée (ligne 3). Ici, le point subtil est que CECA recoure à l'opérateur $\Diamond$ puisque la top clause fournie à DeCA$_{LR}$ doit être dans le langage du pair interrogé, ce qui n'est pas le cas d'un mapping. Pour chaque impliqué dans le langage cible de $\mathcal{P}_k$, CECA^k teste (par réfutation) si cet impliqué est aussi une conséquence de $\mathcal{T}(\mathcal{P}_k)$ (ligne 4). Lorsque CECA est utilisé pour résoudre notre problème de décision, CECA^k s'arrête en répondant No sitôt qu'il trouve un impliqué témoignant de la non conservativité du pair $\mathcal{P}_k$, ou en répondant Yes dans le cas contraire.

Lorsque CECA est utilisé pour résoudre notre problème fonctionnel, CECA^k collecte et retourne les témoins de non conservativité de $\mathcal{P}_k$. Notons que CECA peut être facilement modifié afin de retourner des couples formés d'un témoin et du mapping à partir duquel ce témoin est dérivé (c.-à-d. sa cause). Dans ce cas, la ligne 8 devient $temoins = temoins \cup \{(l_k^1 \vee \cdots \vee l_k^n, l_k \vee l_j)\}$. Ceci est utile pour un pair dans le cas où il veut supprimer les mappings produisant des témoins indésirables.

Algorithme 2: CECA sur le pair $\mathcal{P}_k$ du P2PIS $\mathcal{S}$

CECA^k(dp)
Entrée: un booléen dp
Sortie:
- lorsque $dp = vrai$: YES si $\mathcal{S}$ est une extension conservative de $\mathcal{P}_k$, NO sinon (problème de décision)
- lorsque $dp = faux$: l'ensemble de témoins de la non conservativité de $\mathcal{P}_k$ (problème fonctionnel)

(1) $temoins = \emptyset$
(2) **foreach** clause $l_k \vee l_j \in \mathcal{M}(\mathcal{P}_k)$
(3) **foreach** $l_k^1 \vee \cdots \vee l_k^n \in$ DECA$_{LR}{}^k(l_k, \emptyset, k)$ ⩒ DECA$_{LR}{}^j(l_j, \emptyset, k)$ t.q. $l_k^1 \vee \cdots \vee l_k^n \in Tgt(\mathcal{L}(\mathcal{P}_k))$
(4) **if** $\mathcal{T}(\mathcal{P}) \cup \{\bar{l}_k^1, \ldots, \bar{l}_k^n\} \not\vdash_R \square$
(5) **if** dp
(6) **return** NO
(7) **else**
(8) $temoins = temoins \cup \{l_k^1 \vee \cdots \vee l_k^n\}$
(9) **if** dp
(10) **return** YES
(11) **else**
(12) **return** $temoins$

Le Théorème 4 établit que CECA est une procédure effective pour répondre à nos deux problèmes. La preuve de ce théorème découle du Théorème 2, du Théorème 3 et de la Proposition 3 (pour la ligne 3).

Théorème 4 *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}_k$.*

- CECA$^k(true)$ *retourne* YES *ssi $\mathcal{S}$ est une extension conservative de $\mathcal{P}_k$.*
- CECA$^k(faux)$ *retourne un ensemble de témoins de la non conservativité de $\mathcal{P}_k$, dont tous les impliqués premiers découlant nécessairement des mappings de $\mathcal{P}_k$.*

Exemple (suite) Nous avons mentionné précédemment que le P2PIS de la Figure 1 n'est ni une extension conservative de $\mathcal{P}_1$, ni de $\mathcal{P}_2$, alors que c'est une extension conservative de $\mathcal{P}_3$. Montrons comment $\mathcal{P}_1$, $\mathcal{P}_2$ et $\mathcal{P}_3$ peuvent le savoir à l'aide de CECA.

La non conservativité de $\mathcal{P}_1$ est exhibée lorsque le mapping $\neg L_1 \vee RES_2$ est traité. En effet, on peut facilement vérifier que les impliqués dans le langage cible de $\mathcal{P}_1$ et résultant de DECA$_{LR}{}^1(\neg L_1, \emptyset, 1)$ ⩒ DECA$_{LR}{}^2(RES_2, \emptyset, 1)$ sont $\neg L_1$, $\neg C_1 \vee M_1$ et $\neg C_1 \vee M_1 \vee \neg L_1$. La réfutation de ceux-ci dans $\mathcal{T}(\mathcal{P}_1)$ ne menant pas à $\square$, CECA$^1(vrai)$ s'arrête en retournant NO et CECA$^1(faux)$ retourne les impliqués mentionnés ci-dessus comme témoins de la non conservativité de $\mathcal{P}_1$.

La non conservativité de $\mathcal{P}_2$ est établie quand $\neg M_1 \vee FS_2$ ou $\neg ST_2 \vee PhDS_3$ est traité. En effet, à la fois DECA$_{LR}{}^1(\neg M_1, \emptyset, 2)$ ⩒ DECA$_{LR}{}^2(FS_2, \emptyset, 2)$ et DECA$_{LR}{}^2(\neg ST_2, \emptyset, 2)$ ⩒ DECA$_{LR}{}^3(PhDS_3, \emptyset, 2)$ produisent $\neg ST_2 \vee FS_2$ et $\neg LN_2 \vee RES_2 \vee FS_2$ comme témoins de la non conservativité de $\mathcal{P}_2$. CECA$^2(vrai)$ retourne alors NO et CECA$^2(faux)$ retourne les témoins mentionnés ci-dessus.

Enfin, le P2PIS est une extension conservative de $\mathcal{P}_3$ car il n'y a aucun moyen de dériver un témoin dans le langage cible de $\mathcal{P}_3$ à partir de ses mappings. CECA$^3(vrai)$ s'arrête donc en retournant YES et CECA$^3(vrai)$ retourne un ensemble vide de témoins. ■

Comme expliqué en Section 4, la Proposition 1 et la Proposition 2, et par conséquent la reformulation de l'extension conservative du théorème 2 et le Théorème 4, sont valides *uniquement* pour des P2PIS satisfiables. Toutefois, ceci n'est pas un réel problème. En effet, nous avons ici tous les outils nécessaires pour élaborer des stratégies décentralisées auto-guérissantes de sorte à ce que nous puissions supposer que tout P2PIS est satisfiable.

Tout d'abord, nous pouvons supposer que tout pair d'un P2PIS est satisfiable, puisqu'il peut tester lui-même si sa théorie permet de dériver $\square$ (par ex. en utilisant $\vdash_R$). Maintenant, si un P2PIS est insatisfiable alors que chacun de ses pairs est satisfiable, un mapping est nécessairement impliqué dans toute preuve d'insatisfiabilité. Puisque la déduction linéaire est complète, l'insatisfiabilité d'un P2PIS peut être prouvée par des déductions linéaires dans lesquelles au moins un mapping est impliqué. En réutilisant en partie la preuve de la Proposition 2[8], nous obtenons que tout pair ayant un mapping impliqué dans l'insatisfiabilité d'un P2PIS peut détecter cette insatisfiabilité et aussi la réparer. Pour la détecter, tout pair $\mathcal{P}_k$ peut tester pour chacun de ses mappings $l_k \vee l_j$ si $\square \in \text{DECA}_{LR}{}^k(l_k, \emptyset, k) \veebar \text{DECA}_{LR}{}^j(l_j, \emptyset, k)$ (Proposition 3 et Théorème 3). Si c'est le cas, il peut réparer – certes naïvement – en retirant le mapping $l_k \vee l_j$ de $\mathcal{M}(\mathcal{P}_k)$ et $\mathcal{M}(\mathcal{P}_j)$. Si chaque pair suit une telle stratégie, toutes les preuves d'insatisfiabilité du P2PIS sont "cassées" et le P2PIS est satisfiable. Bien évidemment, des stratégies auto-guérissantes plus sophistiquées peuvent être élaborées (par ex. garantissant d'une façon ou d'une autre qu'un nombre optimal de mappings est supprimé ou encore que seuls les mappings les moins "sûrs" sont supprimés), mais ceci sort du cadre de cet article.

[8]Quand un mapping est utilisé comme side clause dans une déduction linéaire d'un impliqué, il existe une autre déduction linéaire de cet impliqué dans laquelle ce mapping est la top clause.

6 Travaux connexes

À notre connaissance, cet article est le premier à s'intéresser à la notion d'extension conservative dans le cadre des P2PISs. Par conséquent, nous ne pointons que brièvement les quelques travaux de la littérature en lien avec cette notion dans des cadres logiques distribués.

Dans [7, 8], les auteurs revisitent les P2PIS relationnels de [5] pour lesquels il a été montré que répondre à une requête est indécidable, sauf sous certaines contraintes topologiques sévères (acyclicité des mappings). Ils adoptent une sémantique épistémique du premier ordre pour ces P2PIS (en utilisant la logique modale $S5$) de sorte que répondre à une requête est toujours décidable (c.-à-d. même en présence de cycles dans les mappings). On notera que cette sémantique impose à tout mapping de ne plus être une règle d'intégration entre des connaissances de deux pairs, mais simplement une règle d'échange de connaissances entre un pair source et un pair cible : les mappings deviennent unidirectionnels. Au passage, les auteurs de [7] précisent que cette nouvelle sémantique est "meilleure" que la sémantique classique du point de vue de la notion de *modularité* (c.-à-d. d'extention conservative) :

l'interprétation de la KB d'un pair peut différer avec les deux sémantiques lorsque celle-ci est considérée seule ou au sein du P2PIS, mais elle diffère moins avec la sémantique épistémique. Notons que même si nous conservons la sémantique standard de PL, nos résultats peuvent être utilisés pour que l'interprétation de la KB d'un pair ne change pas ou change "juste ce qu'il faut" lorsque elle est considérée seule ou au sein du P2PIS (en éliminant les mappings causant des témoins indésirables de non conservativité du pair).

Peu de travaux ont abordé les P2PISs insatisfiables, un cas particulier de non conservativité de tous les pairs.
Dans [9, 12], les auteurs étudient comment répondre à une requête dans un P2PIS relationnel insatisfiable. Ils recourent à une sémantique épistémique (en utilisant la logique multimodale non monotone $K45_m^{\mathcal{A}}$) dans le but de définir des mappings qui ne propagent à d'autres pairs, ni l'insatisfiabilité locale à un pair, ni l'insatisfiabilité globale du P2PIS. Notons que cette approche étend [7] (voir ci-dessus), et donc qu'elle n'exige aucune contrainte topologique pour répondre à des requêtes.
Dans [11, 43], les auteurs étudient le même problème de raisonnement mais sans adopter une approche épistémique. Ils imposent alors des contraintes topologiques (acyclicité des mappings). Ils ajoutent des relations de confiance entre couples de pairs et recourent à l'*answer set programming* afin de calculer les réponses consistantes d'une requête.
Dans [2], les auteurs étudient le problème de calcul de conséquences dans les P2PIS propositionnels insatisfiables. Leur travail s'appuie sur la notion de *conséquences bien fondées*, c'est-à-dire des conséquences dont une preuve utilise un sous-ensemble satisfiable de clauses d'un P2PIS.
Enfin, les auteurs de [44, 45] étudient aussi le problème de calcul de conséquences dans les P2PISs propositionnels insatisfiables, dans le but de répondre à des requêtes par réfutation. Ils observent que, dans [2], à la fois une formule et sa négation peuvent être des conséquences bien fondées d'un P2PIS. Ils recourent alors à des techniques d'argumentation issues de l'IA, afin de calculer les conséquences ayant de meilleurs raisons d'être vérifiées que leurs négations.
Au lieu de raisonner dans des P2PIS insatisfiables, nous adoptons ici une stratégie décentralisée auto-guérissante de sorte à ce que nos P2PIS soient toujours satisfiables. Intuitivement, ceci revient à supprimer automatiquement les mappings qui contredisent les connaissances propres aux pairs.

7 Conclusion

Nous avons montré en quoi la notion d'extension non conservative d'une KB est importante dans les P2PIS. En particulier, cette notion est étroitement liée à la confidentialité des connaissances d'un pair au sein d'un P2PIS et à la qualité de service des raisonnements proposés par un P2PIS. Nous avons donc étudié comment résoudre *de façon effective*, dans des P2PIS passant à l'échelle d'un millier de pairs, les problèmes suivants : (i) décider si un P2PIS est une extension conservative d'un pair donné et (ii) calculer les témoins (et leurs causes) de la non conservativité d'un pair au sein d'un P2PIS. Une prochaine étape de ce travail est de développer la notion de

stratégie auto-guérissante décentralisée introduite dans cet article.

Il est important de remarquer que nos résultats permettent la résolution de nos deux problèmes dans des P2PIS du Web Sémantique. En effet, il découle aisément des logiques de description utilisées dans les P2PIS SomeOWL [1] et SomeRDFS [15] – des fragments propositionnels de recommandations du W3C pour le Web Sémantique – que nos deux problèmes peuvent être résolus *de facto* dans de tels systèmes en les résolvant dans les P2PIS propositionnels correspondant.

Enfin, nos résultats peuvent aussi être utiles à la découverte automatique de mappings entre ontologies (par ex. [46, 47, 48, 49, 50]) : de nombreux travaux récents ont étudié comment des techniques d'alignement de schémas ou d'ontologies peuvent être utilisées dans les P2PIS du Web Sémantique afin de générer de nouveaux mappings entre pairs (par ex. [51, 52, 51, 53, 54]). Dans de telles approches, il pourrait être intéressant d'exhiber les témoins de non conservativité d'un pair résultant de mappings générés automatiquement. En faisant cela, un expert pourrait choisir les mappings acceptables parmi les mappings générés.

Références

[1] 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 (JAIR)*, 25, 2006.

[2] 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 (ECAI)*, 2006.

[3] L. Serafini, A. Borgida, and A. Tamilin. Aspects of distributed and modular ontology reasoning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[4] L. Serafini and A. Tamilin. DRAGO : Distributed reasoning architecture for the semantic web. In *European Semantic Web Conference (ESWC)*, 2005.

[5] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *International Conference on Data Engineering (ICDE)*, 2003.

[6] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *SIGMOD Record*, 32(3), 2003.

[7] 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 (PODS)*, 2004.

[8] 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 (VLDB)*, 2004.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in p2p

data integration : an epistemic logic approach. In *International Symposium on Database Programming Languages (DBPL)*, 2005.

[10] L. Caroprese, C. Molinaro, and E. Zumpano. Integrating and querying p2p deductive databases. In *International Database Engineering and Applications Symposium (IDEAS)*, 2006.

[11] L. E. Bertossi and L. Bravo. The semantics of consistency and trust in peer data exchange systems. In *International Conferences on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, 2007.

[12] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in p2p data integration : An epistemic logic approach. *Information Systems (IS)*, 33(4-5), 2008.

[13] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella : a p2p networking infrastructure based on rdf. In *International World Wide Web Conference (WWW)*, 2002.

[14] G. Tummarello, C. Morbidoni, and M. Nucci. Enabling semantic web communities with dbin : An overview. In *International Semantic Web Conference (ISWC)*, 2006.

[15] P. Adjiman, F. Goasdoué, and M.-C. Rousset. Somerdfs in the semantic web. *Journal on Data Semantics (JODS)*, 8, 2007.

[16] S. Ghilardi, C. Lutz, and F. Wolter. Did i damage my ontology ? a case for conservative extensions in description logics. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2006.

[17] C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[18] C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic EL. In *Conference on Automated Deduction (CADE)*, 2007.

[19] R. Kontchakov, F. Wolter, and M. Zakharyaschev. Modularity in DL-lite. In *International Workshop on Description Logics (DL)*, 2007.

[20] S. Ghilardi, C. Lutz, F. Wolter, and M. Zakharyaschev. Conservative extensions in modal logics. In *Advances in Modal Logics (AiML)*, volume 6. College Publications, 2006.

[21] G. Antoniou and A. Kehagias. A note on the refinement of ontologies. *International Journal of Intelligent Systems (JIS)*, 15, 2000.

[22] A. Herzig and I. J. Varzinczak. A modularity approach for a fragment of *lc*. In *European Conference on Logics in Artificial Intelligence (JELIA)*, 2006.

[23] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and web ontologies. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2006.

[24] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount : extracting modules from ontologies. In

International World Wide Web Conference (WWW), 2007.

[25] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. A logical framework for modularity of ontologies. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[26] M.-C. Rousset. Small can be beautiful in the semantic web. In *International Semantic Web Conference (ISWC)*, 2004.

[27] S. A. Cook. The complexity of theorem proving procedures. In *Symposium on Theory of Computing (STOC)*, 1971.

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

[29] J. Hladik and J. Model. Tableau systems for shio and shiq. In *International Workshop on Description Logics (DL)*, 2004.

[30] 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 (IJCAI)*, 2005.

[31] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook : Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[32] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics : The *dl-lite* family. *Journal of Automated Reasoning (JAR)*, 39(3) :385–429, 2007.

[33] W. Craig. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic (JSL)*, 22(3) :269–285, 1957.

[34] T. Dimitrakos and T. S. E. Maibaum. Notes on refinement, interpolation and uniformity. In *International Conference on Automated Software Engineering (ASE)*, 1997.

[35] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1), 1965.

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

[37] E. Minicozzi and R. Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence (AI)*, 3(1-3), 1972.

[38] K. Inoue. Consequence-finding based on ordered linear resolution. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1991.

[39] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence (AI)*, 2-3(56), 1992.

[40] E. Amir and S. McIlraith. Partition-based logical reasoning. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2000.

[41] E. Amir and S. McIlraith. Theorem proving with structured theories. In

International Joint Conference on Artificial Intelligence (IJCAI), 2001.

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

[43] L. E. Bertossi and L. Bravo. Information sharing agents in a peer data exchange system. In *International Conference on Data Management in Grid and P2P Systems (Globe)*, 2008.

[44] A. Binas and S. McIlraith. Exploiting prefereces over information sources to efficiently resolve inconsistencies in peer-to-peer query answering. In *AAAI Workshop on Preference Handling for Artificial Intelligence*, 2007.

[45] A. Binas and S. McIlraith. Peer-to-peer query answering with inconsistent knowledge. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2008.

[46] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4), 2001.

[47] Y. Kalfoglou and M. Schorlemmer. Ontology mapping : the state of the art. *The Knowledge Engineering Review*, 18(1), 2003.

[48] N. F. Noy. Semantic integration : A survey of ontology-based approaches. *SIGMOD Record*, 33(4), 2004.

[49] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics (JODS)*, 4, 2005.

[50] A. Doan and A. Halevy. Semantic integration research in the database community : A brief survey. *AI Magazine*, 26(1), 2005.

[51] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Y. Halevy. Learning to match ontologies on the semantic web. *VLDB Journal*, 12(4), 2003.

[52] S. Castano, A. Ferrara, and S. Montanelli. H-match : an algorithm for dynamically matching ontologies in peer-based systems. In *Semantic Web and DataBases Workshop (SWDB)*, 2003.

[53] S. Herschel and R. Heese. Humboldt discoverer : A semantic p2p index for pdms. In *International Workshop Data Integration and the Semantic Web (DISWeb)*, 2005.

[54] F.-E. Calvier and C. Reynaud. Ontology matching supported by query answering in a p2p system. In *International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, 2008.