# Calcul de conséquences pour le test d'extension conservative dans un système pair-à-pair

Nada Abdallah, François Goasdoué

▶ **To cite this version:**

Nada Abdallah, François Goasdoué. Calcul de conséquences pour le test d'extension conservative dans un système pair-à-pair. JFPC 2008- Quatrièmes Journées Francophones de Programmation par Contraintes, LINA - Université de Nantes - Ecole des Mines de Nantes, Jun 2008, Nantes, France. pp.95-103. inria-00291089

**HAL Id: inria-00291089**
**https://inria.hal.science/inria-00291089**

Submitted on 26 Jun 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Calcul de conséquences pour le test d'extension conservative dans un système pair-à-pair

**Nada Abdallah** **François Goasdoué**

Univ. Paris-Sud & CNRS (LRI/IASI) – INRIA (Saclay – Île-de-France/GEMO)
LRI, Bâtiment 490, Univ. Paris-Sud, 91405 Orsay Cedex, France
{nada,fg}@lri.fr

**Résumé**

Dans un système d'inférence pair-à-pair (P2PIS), un pair étend sa base de connaissances (KB) avec celle des autres pairs afin d'utiliser leurs connaissances pour répondre aux requêtes qui lui sont posées. Toutefois, l'extension d'une KB n'est pas nécessairement conservative. Une extension conservative garantit que le sens d'une KB est le même lorsqu'elle est considérée seule ou avec son extension. En revanche, une extension non conservative peut changer radicalement le sens d'une KB au sein de la théorie résultante. Il est par conséquent crucial pour un pair de savoir si un P2PIS est une extension conservative de sa KB. En effet, si ce n'est pas le cas (i) ses utilisateurs n'ont plus la bonne interprétation de sa KB, donc des requêtes en termes de sa KB et des réponses à celles-ci et (ii) les connaissances qu'il fournit aux autres pairs ne sont pas celles escomptées.
Cet article est le premier à s'intéresser à la notion d'extension conservative dans le cadre des P2PIS. Notre contribution est d'étudier théoriquement et algorithmiquement le problème de tester si un P2PIS propositionnel est une extension conservative d'un pair donné. En particulier, nous recourons au calcul de conséquences afin de reformuler ce problème et d'élaborer un algorithme décentralisé pour le résoudre.

**Abstract**

In a peer-to-peer inference system (P2PIS), a peer basically extends its knowledge base (KB) with those of the other peers so that it can also use their knowledge to answer queries. However, the extension of a KB is not necessarily conservative. A conservative extension guarantees that the meaning of a KB remains the same when it is considered alone or together with its extension. In contrast, a non-conservative extension may radically change the meaning of a KB within the resulting logical theory. It is therefore crucial to a peer to know whether a P2PIS is a conservative extension of its KB since non-conservativeness indicates that (i) its users do not have anymore the right meaning of its KB, thus of queries w.r.t. that KB, so of the answers to queries and (ii) the knowledge it provides to the other peers is not the expected one.
This paper is the first to address conservative extension in the setting of P2PISs. Our contribution is to study the problem of checking whether a propositional P2PIS is a conservative extension of a given peer from both the theoretical and algorithmic perspectives. In particular, we resort to consequence finding to reformulate that problem and thus to devise a fully decentralized algorithm that solves it.

## 1 Introduction

Ces dernières années, les *systèmes d'inférence pair-à-pair* (P2PIS) ont reçu une attention considérable car leur infrastructure permet le raisonnement sur des connaissances disséminées dans des réseaux. Par exemple, de nombreuses tâches d'inférence fondamentales en *Intelligence Artificielle* et en *Bases de Données* ont été étudiées dans ce cadre distribué, notamment le calcul de conséquences en logique propositionnelle (*eg.* [1, 2, 3, 9]), le test de subsomption en logique de description (*eg.* [25, 24]), ou encore répondre à une requête dans des langages relationnels ou du Web Sémantique (*eg.* [27, 17, 8, 11, 7, 3, 4]).
Les P2PIS sont des systèmes fondés sur la logique et constitués de serveurs autonomes (*ie.* conçus et administrés indépendamment les uns des autres) appelés *pairs*. Ces pairs partagent des connaissances en utilisant des formules particulières appelées *mappings* qui établissent des correspondances sémantiques entre leurs bases de connaissances (KB). De cette manière, une tâche d'inférence initiée sur la KB d'un pair donné peut être effectuée sur la KB globale du P2PIS (*ie.* l'union des KB et mappings de tous les pairs) en se propageant de proche en proche dans le système via les mappings appropriés.
Toutefois, raisonner dans un P2PIS n'est pas si simple. En effet, 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. Ce cadre distribué impose par conséquent d'élaborer des algorithmes *décentralisés* de raisonnement qui doivent terminer et être corrects et complets pour les tâches d'inférence sur la KB du P2PIS, *sans* avoir accès à cette connaissance globale.

Cet article est le premier à étudier la notion d'*extension conservative d'une KB* dans le cadre des P2PIS. Formellement, étant données deux KB $\Sigma_1$ et $\Sigma_2$ utilisant la même logique, on dit que $\Sigma_1 \cup \Sigma_2$ est une extension conservative de $\Sigma_1$ si et seulement si toute conséquence logique de $\Sigma_1 \cup \Sigma_2$, utilisant seulement (un sous ensemble donné) des symboles de $\Sigma_1$, est aussi une conséquence de $\Sigma_1$.
Cette notion s'est déjà révélée utile dans des cadres logiques centralisés pour la maintenance, la réutilisation et l'intégration de KB. Par exemple, il est parfois important de garantir que certaines sortes de mises-à-jour de KB (*eg.* raffiner une KB ou fusionner une KB avec une autre) préserve la signification de la KB originale [5] (*eg.* pour utiliser la mise-à-jour à la place de la KB originale dans des applications critiques). Jusqu'à présent, la notion d'extension conservative dans des cadres centralisés a été principalement étudiée en logique de description (*eg.* [12, 16, 19, 15, 14, 20, 18]) et en logique modale (*eg.* [13]).
La notion d'extension conservative est aussi pertinente dans le cadre distribué des P2PIS : tout pair d'un P2PIS étend sa KB avec celle des autres pairs de sorte à utiliser leurs connaissances pour répondre aux requêtes qui lui sont posées. Il est même *crucial* pour un pair de savoir si un P2PIS est une extension conservative de sa KB. En effet, si ce n'est pas le cas (i) ses utilisateurs n'ont plus la bonne interprétation de sa KB, donc des requêtes en termes de sa KB et des réponses à celles-ci et (ii) les connaissances qu'il fournit aux autres pairs ne sont pas celles escomptées.

Nous étudions ici, d'un point de vue théorique et algorithmique, le problème consistant à *decider si un P2PIS propositionnel de [1, 2, 3] est une extension conservative d'un pair donné*. On notera qu'étudier ce problème dans de tels P2PIS est d'un grand intérêt puisque non seulement ils proposent la tâche fondamentale de calcul de conséquences dans une KB (distribuée) de la logique propositionnelle, mais aussi car des P2PIS utilisant d'autres logiques et proposant d'autres tâches de raisonnement sont en fait construits au-dessus de ces P2PIS propositionnels et utilisent le calcul de conséquences pour mettre en oeuvre leurs propres tâches de raisonnement. Par exemple, c'est le cas des P2PIS *SomeOWL* (aussi connu sous le nom de *SomeWhere*) [3] et *SomeRDFS* [4] pour le Web Sémantique qui offrent la tâche centrale de Bases de données consistant à répondre à une requête.
Notre contribution est d'établir la complexité exacte du problème de test d'extension conservative d'un pair, puis d'exhiber une reformulation de ce problème en utilisant des algorithmes centralisés de calcul de conséquences bien connus. En adaptant ensuite ce calcul à notre cadre distribué, nous élaborons un algorithme totalement décentralisé pour notre problème. Cet algorithme s'exécute sur chacun des pairs et permet à chacun d'eux de vérifier, à tout instant, si le P2PIS est une extension conservative de ses propres connaissances.
L'article est organisé comme suit. Nous définissons les P2PIS que nous considérons dans la Section 2. Nous présentons notre problème d'extension conservative, ainsi que sa complexité, dans la Section 3. Nous reformulons ce problème en terme de calcul de conséquences dans la Section 4. Dans la Section 5, nous fournissons notre algorithme de test d'extension conservative (CECA) fondée sur cette reformulation. 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 Système d'inférence pair-à-pair

Un P2PIS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ est un ensemble de pairs.

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. On notera que les alphabets des pairs sont disjoints. Nous supposons que chaque pair a un unique identifiant (*eg.* son adresse IP) et que le nom de ses variables utilisent cet identifiant d'une manière ou d'une autre. Ici, nous utilisons l'indice $i$ comme identifiant du pair et notons une variable $A$ du pair $\mathcal{P}_i$ par $A_i$. 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, *eg.* ici pour tester l'extension conservative seulement par rapport à un sous-langage donné. 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 $\mathcal{M}(\mathcal{P}_i)$, *ie.* tout mapping entre deux pairs est stocké dans ces deux pairs. Chaque mapping entre les pairs $\mathcal{P}_i$ et $\mathcal{P}_j$ est de la forme $l_i \vee l_j$, tel que $i \neq j$ et $l_i \in \mathcal{L}(\mathcal{P}_i)$ et $l_j \in \mathcal{L}(\mathcal{P}_j)$ sont des littéraux. 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.

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 $\mathcal{P}_3$ 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. ■

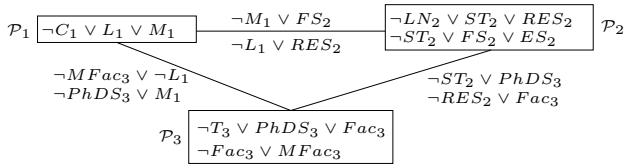


FIG. 1 – Le P2PIS $\mathcal{S}$.

## 3 Test d'extension conservative

Nous étudions ici le problème consistant à *décider si un P2PIS est une extension conservative d'un pair donné*. Tout d'abord, nous définissons formellement la notion d'*extension conservative d'un pair dans un P2PIS*, celle-ci se fondant sur la notion *conséquence* logique. Nous posons ensuite notre problème d'extension conservative, puis nous établissons sa complexité.

**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 d'un pair)** *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$. $\mathcal{S}$ est une* extension conservative *de $\mathcal{P}$ ssi la propriété $\Psi$ est vérifiée.*
*$\Psi$ : pour tout $c \in Tgt(\mathcal{L}(\mathcal{P}))$, si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ alors $\mathcal{T}(\mathcal{P}) \models c$.*

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

**Définition 3 (Problème)**
DONNÉES : *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$.*
QUESTION : *Est-ce que pour tout $c \in Tgt(\mathcal{L}(\mathcal{P}))$, si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ alors $\mathcal{T}(\mathcal{P}) \models c$ ?*

Il est relativement aisé de voir – à partir de la Définition 3 – que notre problème est difficile. Il consiste en effet à résoudre des problèmes coNP-complets[1] pour un nombre exponentiel de clauses[2] (dans le pire des cas). Cette intuition est confirmée par le théorème suivant qui établit la complexité exacte de notre problème.

**Théorème 1 (Complexité du problème)** *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}$. Décider si $\mathcal{S}$ est une extension conservative de $\mathcal{P}$ est $\Pi_2^p$-complet.*

Ce problème est $\Pi_2^p$-facile car le problème complémentaire est dans $\Sigma_2^p$. 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 $\Pi_2^p$-difficile du fait d'une réduction polynomiale depuis le problème de validité d'une formule $QBF_{2,\forall}$, le problème canonique de la classe $\Pi_2^p$. 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 $\Phi$. Puisque $\forall A \exists B \Phi$ est valide si et seulement si il n'existe pas de conséquence logique de $\Phi$ exprimée uniquement en termes de variables de $A$ [21], une preuve de $\Pi_2^p$-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 $\Phi$ exprimée en termes de $A$. Par exemple, puisque $\Phi$ 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 (*ie.* il n'y a pas de conséquence de $\Phi$ exprimée uniquement en termes de $A$) si et seulement si ce P2PIS est une extension conservative de $\mathcal{P}_1$.

**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 $\mathcal{P}_3$.)
Par exemple, on peut facilement vérifier (*eg.* 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 $\mathcal{T}(\mathcal{P}_2) \not\models \neg ST_2 \vee FS_2$. On notera 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 ! ■

[1] Décider $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$, resp. $\mathcal{T}(\mathcal{P}) \models c$, revient à décider $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{\neg c\} \models \square$, resp. $\mathcal{T}(\mathcal{P}) \cup \{\neg c\} \models \square$, *ie.* à UNSAT.

[2] 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 $\mathcal{A}(\mathcal{P})$.

## 4 Test d'extension conservative d'un pair et calcul de conséquences

Afin de décider si un P2PIS est une extension conservative d'un pair donné, nous allons calculer des conséquences du P2PIS exprimées dans le langage cible de ce pair, puis tester si ces conséquences sont aussi des conséquences du pair seul. Évidemment, nous garantissons que l'ensemble des conséquences calculées est toujours suffisant pour répondre correctement à notre problème de décision.

Laissons de côté – pour le moment – les aspects distribués de notre cadre logique et montrons comment résoudre notre problème en utilisant des outils centralisés bien connus de calcul de conséquences. Nous recourons à la notion d'*impliqués premiers d'une théorie propositionnelle* (*ie.* ses conséquences les plus fortes) afin de fournir une caractérisation *exacte* des conséquences à calculer.

**Définition 4 (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 $f \models f'$ (*ie. *$f' \equiv f$).*

Notons que la résolution [23] 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 [6] qui sont correctes et complètes pour le calcul de conséquences [22, 21]. 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*[3] (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[4] [21]. Par conséquent, il découle de la Définition 4 que trouver l'ensemble des impliqués premiers d'un P2PIS exprimés dans le langage cible d'un pair donné est *suffisant* afin de décider l'extension conservative pour ce pair. Toutefois, il n'est *pas nécessaire* de trouver tous ces impliqués. 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 (*ie.* il satisfait alors nécessairement la propriété $\Psi$ de la Définition 3), soit il n'est pas exprimé dans le language (cible) de ce pair (*ie.* il n'a alors rien à voir avec la propriété $\Psi$ 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 pour 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 lorsque 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 $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^{m}_{LR} c$.*

On remarquera 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 $\Box$, qui appartient au langage cible de tous les pairs. Par conséquent, ces propositions seraient vérifiées si $\Box$ pouvait être dérivée (par déduction linéaire) à partir d'un mapping de chaque pair, ce qui n'est – bien évidemment – pas le cas (*eg.* 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 à la reformulation suivante de la Définition 3 en termes d'outils de calcul (centralisé) de conséquences. Au passage, on remarquera que la Définition 5 fournit aussi une manière effective de tester l'extension conservative pour des KB propositionnelles ou qui peuvent être compilées en KB propositionnelles, *eg.* des ontologies écrites en OWL-PL [3] (le fragment $\mathcal{CLU}$ de la logique de description OWL-DL, une recommandation du W3C pour construire des ontologies complexes) ou en coreRDFS [4] (le fragment de logique de description de RDFS, une recommandation du W3C pour construire des ontologies simples).

**Définition 5 (Reformulation du problème)**
Données : *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}$.*
Question : *Est-ce que pour tout $m \in \mathcal{M}(\mathcal{P})$, si $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^{m}_{LR} c$ avec $c \in Tgt(\mathcal{L}(\mathcal{P}))$ alors $\mathcal{T}(\mathcal{P}) \cup \{\neg c\} \vdash_R \Box$ ?*

Si nous considérons maintenant les aspects distribués de notre cadre logique, il découle de la définition ci-dessus que nous avons besoin d'un algorithme *décentralisé* de déduction linéaire dans un P2PIS afin d'obtenir une procédure de décision effective pour notre problème de test d'extension conservative.

## 5 Algorithme décentralisé pour le test d'extension conservative d'un pair

Élaborons maintenant un algorithme décentralisé pour décider si un P2PIS est une extension conservative d'un pair donné. Nous proposons tout d'abord un algorithme décentralisé de calcul de conséquences fondé sur la résolution linéaire (DeCA$_{LR}$, Algorithme 1), puis nous présentons notre algorithme de test d'extension conservative d'un pair (CECA, Algorithme 2) qui recoure à DeCA$_{LR}$ en suivant la Définition 5.

DeCA$_{LR}$ **en bref** DeCA$_{LR}$ est une adaptation de l'algorithme décentralisé DeCA qui avait été proposé pour trouver les impliqués premiers *propres* dans les P2PIS de [1, 2, 3] (qui sont similaires à ceux de cet article). En effet, DeCA a été conçu pour calculer des impliqués d'un P2PIS *et* d'une clause donnée exprimée

[3] Dans la suite, nous utilisons la terminologie de [6] pour parler de la déduction linéraire (*eg.* top clause, center clause ou encore side clause).

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

dans le langage d'un pair, en particulier *tous* les premiers qui sont *propres* à cette clause : les impliqués premiers du P2PIS *et* de cette clause qui ne sont pas des impliqués premiers du P2PIS *seul* (*ie.* sans y ajouter cette clause). On notera que les impliqués calculés par DeCA sont obtenus par *déductions* (décentralisées) et peuvent être exprimées en termes de variables cibles de différents pairs. En revanche, DeCA$_{LR}$ est conçu pour calculer des impliqués d'un P2PIS *et* d'une clause donnée exprimée dans le langage d'un pair, notamment *tous* les premiers pouvant être dérivés à partir de cette clause par *déductions linéaires* (décentralisées). De plus, les impliqués calculés sont dans le langage cible d'un *unique* pair et ne sont *pas nécessairement propres* à la clause fournie en entrée : nous voulons utiliser DeCA$_{LR}$ dans la Définition 5 à la place de la déduction linéaire et l'objectif de cette tâche d'inférence n'est pas la dérivation des seuls impliqués premiers propres.
Afin d'obtenir DeCA$_{LR}$ à partir de DeCA, nous avons donc changé les capacités de raisonnement local des pairs (déduction linéaire au lieu de déduction), les conditions d'arrêt puisque nous ne nous focalisons pas sur les impliqués (premiers) *propres* et les conditions de filtrage afin de produire uniquement des impliqués dans le langage cible d'*un* pair donné.

DeCA$_{LR}$ étant une adaptation de DeCA, il suit le même schéma algorithmique. C'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. Comme dans DeCA, 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 puisqu'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 prévenir les récursions infinies qui arrivent lorsqu'un même pair est sollicité plus d'une fois pour une *même* clause au sein d'appels récursifs imbriqués.

Étant donné 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 $q_k$ (line 4).

Dans un second temps (ligne 10), DeCA$_{LR}{}^k$ calcule de nouveaux impliqués (premiers) à partir de ceux obtenus dans la première étape en utilisant l'opérateur de distribution clausale $\otimes$[5]. DeCA$_{LR}{}^k$ commence par utiliser les mappings pour solliciter des pairs (qui exécutent 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 11).
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 12), et 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 14).

**Algorithme 1:** DeCA$_{LR}$ sur le pair $\mathcal{P}_k$ du P2PIS $\mathcal{S}$
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** $q_k \in hist$ (*ie.* $q_k$ 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** $\{\square\}$ % seul $\square$ est premier
(7) **else**
(8) **if** $k \neq id$
(9) retirer de $CI$ toute clause ayant un littéral $l \in \mathcal{L}(\mathcal{P}_k)$ non partagé
(10) **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 $l_k^1 \vee \cdots \vee l_k^q = \square$
(11) $CI = CI \cup \{l_k^1 \vee \cdots \vee l_k^q\} \otimes \otimes_{i=q+1}^{n} [\{l_k^i\} \cup \{c \mid c \in \text{DeCA}_{LR}{}^j(l_j, hist \cup \{q_k\}, id)$ and $\bar{l_k^i} \vee l_j \in \mathcal{M}(\mathcal{P}_k)\}]$
(12) **if** $\square \in CI$
(13) **return** $\{\square\}$ % seul $\square$ est premier
(14) retirer de $CI$ toute clause n'appartenant pas à $\mathcal{L}(\mathcal{P}_{id})$
(15) **return** $CI$

[5] 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 2 énonce les propriétés de DECA$_{LR}$. Il établit que DECA$_{LR}$ est une procédure effective pour la déduction linéaire décentralisée dans le langage d'un pair, dont la top clause est aussi dans le langage de ce pair. Notons que DECA$_{LR}$ peut être facilement modifié de sorte à ce qu'il calcule des impliqués (premiers) en termes de variables (cibles) de différents pairs. Toutefois, ce n'est pas intéressant ici d'un point de vue "optimisation", car l'extension conservative d'un pair est décidée par rapport aux impliqués (premiers) dans le langage (cible) de ce pair. La preuve du Théorème 2 se fonde sur la Proposition 3 (pour la ligne 11).

**Théorème 2** *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* $c \in \mathcal{L}(\mathcal{P}_k)$ *et* $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$, *et*
- *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$_{LR}{}^k(q_k, \emptyset, k)$.

**Proposition 3** *Soit $\mathcal{T}$ une théorie propositionnelle clausale dont une clause est $l_1 \vee \cdots \vee l_n$ :*
- *si* $c \in \Cup_{i=1}^{n} \{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$ *alors* $\mathcal{T} \vdash_{LR}^{l_1 \vee \cdots \vee l_n} c$ *et*
- *si* $\mathcal{T} \vdash_{LR}^{l_1 \vee \cdots \vee l_n} c$ *et* $c \in \mathbf{PI}(\mathcal{T})$ *alors* $c \in \Cup_{i=1}^{n} \{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$.

CECA **en bref** En suivant la Définition 5, CECA$^k$ (CECA sur $\mathcal{P}_k$) considère chaque mapping de $\mathcal{M}(\mathcal{P}_k)$ (ligne 1) à partir duquel il calcule des impliqués (premiers) du P2PIS en utilisant DECA$_{LR}$ comme outil de déduction linéaire décentralisée (ligne 2). Ici, le point subtil est que CECA recoure à $\Cup$ puisque la top clause fournie à DECA$_{LR}$ doit être dans le langage d'*un* pair donné, 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 3). 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. Notons que CECA peut être facilement modifié afin de retourner *tous* les impliqués (premiers) permettant d'établir la non conservativité de l'extension du pair $\mathcal{P}_k$.

Le Théorème 3 établit que CECA est une procédure effective pour décider si un P2PIS est une extension conservative d'un pair donné. La preuve de ce théorème découle de la Définition 5, du Théorème 2 et aussi de la Proposition 3 (pour la ligne 2).

**Théorème 3** *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}_k$.* CECA$^k()$ *retourne* YES *ssi $\mathcal{S}$ est une extension conservative de $\mathcal{P}_k$.*

**Algorithme 2:** CECA sur le pair $\mathcal{P}_k$ du P2PIS $\mathcal{S}$
CECA$^k()$
**Entrée:** Aucune
**Sortie:** YES si $\mathcal{S}$ est une extension conservative de $\mathcal{P}_k$ et NO dans le cas contraire
(1)**foreach** clause $l_k \vee l_j \in \mathcal{M}(\mathcal{P}_k)$
(2) **foreach** $l_k^1 \vee \cdots \vee l_k^n \in$ DECA$_{LR}{}^k(l_k, \emptyset, k) \Cup$ DECA$_{LR}{}^j(l_j, \emptyset, k)$ t.q. $l_k^1 \vee \cdots \vee l_k^n \in Tgt(\mathcal{L}(\mathcal{P}_k))$
(3) **if** $\mathcal{T}(\mathcal{P}) \cup \{\bar{l}_k^1, \ldots, \bar{l}_k^n\} \not\vdash_R \Box$
(4) **return** NO
(5)**return** YES

**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 maintenant 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 CECA$^1$ traite le mapping $\neg L_1 \vee RES_2$. En effet, on peut facilement vérifier que les impliqués dans le langage cible de $\mathcal{P}_1$ et résultant de DECA$_{LR}{}^1(\neg L_1, \emptyset, 1) \Cup$ 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 $\Box$, CECA$^1()$ s'arrête en retournant NO.

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 par CECA$^2$. En effet, à la fois DECA$_{LR}{}^1(\neg M_1, \emptyset, 2) \Cup$ DECA$_{LR}{}^2(FS_2, \emptyset, 2)$ et DECA$_{LR}{}^2(\neg ST_2, \emptyset, 2) \Cup$ 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 impliqués du P2PIS dans le langage cible de $\mathcal{P}_2$. La réfutation de ceux-ci ne menant pas à $\Box$, CECA$^2()$ s'arrête en retournant NO.

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

Comme expliqué en Section 4, la Proposition 1 et la Proposition 2, et par conséquent la reformulation du problème de test d'extension conservative de la Définition 5 et le Théorème 3, 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 autoguérissantes, 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 $\Box$ (*eg.* 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'insatis-

fiabilité. 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[6], nous avons 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 $\square \in \text{DECA}_{LR}{}^{k}(l_k, \emptyset, k) \otimes \text{DECA}_{LR}{}^{j}(l_j, \emptyset, k)$ (Proposition 3 et Théorème 2). 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 plus sophistiquées peuvent être définies, mais ceci sort du cadre de cet article.

## 6 Travaux connexes de la littérature

L'ensemble de travaux [12, 16, 19, 15, 14, 20, 18] fournira au lecteur intéressé une vision assez précise des travaux effectués jusqu'à présent autour de la notion d'extension conservative dans un cadre logique centralisé. Nous pointons ici seulement les quelques travaux sur les P2PIS qui ont un rapport avec le problème de test d'extension conservative étudié dans cet article.

[8, 11] revisitent les P2PIS relationnels de [17] pour lesquels il a été montré que répondre à une requête est indécidable, sauf sous certaines contraintes topologiques sévères (acyclicité des mappings). [8, 11] proposent d'adopter une sémantique épistémique du premier ordre pour ces P2PIS de sorte que répondre à une requête est toujours décidable. Ils précisent aussi que cette nouvelle sémantique est plus adaptée que la sémantique classique si on considère la notion de *modularité* (*ie.* d'extention conservative) : l'interprétation de la KB d'un pair peut être différente avec les deux sémantique lorsqu'on la considère seule ou au sein du P2PIS, mais elle diffère moins avec la sémantique épistémique.

Peu de travaux ont abordé les P2PIS insatisfiables. [7] étudie comment répondre à une requête dans un P2PIS relationnel insatisfiable. Il recoure à une sémantique épistémique dans le but de définir des mappings qui ne propagent ni l'insatisfiabilité locale à un pair, ni l'insatisfiabilité globale du P2PIS, à d'autres pairs.
[9] étudie le calcul de conséquences dans les P2PIS propositionnels insatisfiables de [1, 2, 3]. Ce travail se fonde sur la notion de *conséquences bien fondées* qui sont des conséquences dont une preuve utilise un sous-ensemble satisfiable de clauses d'un P2PIS.
Contrairement aux deux approches ci-dessus, nous adoptons ici une stratégie décentralisée auto-guérissante de sorte à ce que nos P2PIS soient toujours satisfiables.

[6] 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.

## 7 Conclusion & Perspectives

Nous avons étudié le problème consistant à décider si un P2PIS propositionnel est une extension conservative d'un pair donné. Nous avons montré que ce problème est $\Pi_2^p - complet$ et qu'il peut être résolu grâce à un calcul de conséquences décentralisé basé sur la résolution linéaire.
Comme mentionné en introduction de cet article, ce problème est central pour l'Intelligence Artificielle distribuée et les Bases de données distribuées. Il peut aussi être utile à d'autres domaines comme l'Ingénierie des Connaissances puisque, par exemple, l'établissement automatique de mappings entre KB [26, 10] dans un P2PIS peut mener à la non-conservativié de pairs du système.

La prochaine étape de ce travail pourrait être l'étude de stratégies auto-guérissantes sophistiquées en cas de non conservativité d'un pair, et aussi en cas d'insatisfiabilité d'un P2PIS puisqu'il s'agit d'un cas particulier de non conservativité de tous les pairs.

## 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*, 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*, 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*, 2006.

[4] P. Adjiman, F. Goasdoué, and M.-C. Rousset. Somerdfs in the semantic web. *Journal on Data Semantics*, 8, 2006.

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

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

[7] 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*, 2005.

[8] 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*, 2004.

[9] 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*, 2006.

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

[11] 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*, 2004.

[12] 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*, 2006.

[13] S. Ghilardi, C. Lutz, F. Wolter, and M. Zakharyaschev. Conservative extensions in modal logics. In Guido Governatori, Ian Hodkinson, and Yde Venema, editors, *Advances in Modal Logics*, volume 6. College Publications, 2006.

[14] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount : extracting modules from ontologies. In *International World Wide Web Conference*, 2007.

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

[16] 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*, 2006.

[17] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *International Conference on Data Engineering*, 2003.

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

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

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

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

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

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

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

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

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

[27] 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.