Raisonnements standard et non-standard pour les systèmes décentralisés de gestion de données et de connaissances

Nada Abdallah

Thèse

Université Paris Sud 11

présentée pour obtenir le grade de

Docteur de l'Université Paris-Sud 11

Spécialité : Informatique

Par

Nada ABDALLAH

Raisonnements standard et non-standard pour les systèmes décentralisés de gestion de données et de connaissances

Thèse soutenue le 13 Juillet 2010 devant le jury composé de :

M.	Serge Abiteboul, Directeur de recherche, INRIA	Directeur de thèse
M.	Jérôme Euzenat, Directeur de recherche, INRIA	Rapporteur
M.	François Goasdoué, Maître de conférences, Université Paris-Sud 11	Co-directeur de thèse
M.	Mohand-Said Hacid, Professeur, Université Claude Bernard Lyon 1	Rapporteur
M.	Nicolas Spyratos, Professeur, Université Paris-Sud 11	Président

Laboratoire de Recherche en Informatique, INRIA Saclay-île-de-France, U.M.R. CNRS 8623,
Université Paris-Sud, 91405 Orsay Cedex, France

Table des matières

Introduction

Contexte et définition du problème

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).
En IA, des P2PIS ont été étudiés pour le calcul de conséquences (par exemple [7, 8, 31]), le diagnostic à base de modèles (par exemple [15, 14]), et répondre à des requêtes (par exemple [21, 22]) dans des réseaux d'agents intelligents en logique propositionnelle, ou encore le test de subsomption dans des réseaux d'ontologies exprimées en logiques de description (par exemple [63, 64]).
En BD, des P2PIS ont été étudiés pour répondre à des requêtes dans des réseaux de BD relationnelles (par exemple [46, 66, 27, 39, 24, 29, 17, 26]) ou dans le Web Sémantique (par exemple [57, 67, 8, 9]). Ces P2PIS sont plus connus dans la littérature sous le nom de systèmes de gestion de données pair-à-pair (PDMS).

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 (BC) modélisant son domaine d'application ou son champ d'expertise en termes de son *propre* vocabulaire (par exemple 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 BC en utilisant des formules logiques particulières appelées *mappings*. Ces mappings jouent un rôle essentiel puisque, d'une part, ils définissent comment les BC des pairs sont intégrées et, d'autre part, ils donnent lieu à un réseau sémantique dans lequel il devient possible de raisonner (l'union des BC et des mappings de tous les pairs). En effet, une tâche d'inférence sur la BC d'un pair donné peut être effectuée sur la BC 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 BC 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 BC sur laquelle la tâche de raisonnement doit être effectuée (ici la BC 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 BC du P2PIS à un calcul décentralisé entre pairs.

Dans cette thèse, nous considérons les P2PIS propositionnels (c-à-d. où les connaissances des pairs sont décrites par des théories de la logique propositionnelle). Le choix de la logique propositionnelle (LP) poursuit l'idée que "small can be beautiful" [62] afin de fournir des services

de raisonnement utiles, effectifs et passant à l'échelle de grands systèmes. Notre crédo est que LP est *un* bon compromis entre expressivité et efficacité/passage à l'échelle des P2PIS.

Du point de vue de la calculabilité, l'utilisation de LP 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) [8]. 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 PDMS relationnel, à moins de recourir à des contraintes topologiques (acyclicité des mappings) [46] ou encore à des sémantiques non standards (sémantiques épistémiques) [27, 39] avec lesquelles un mapping n'est plus une règle d'intégration entre 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 LP se situent au premier niveau de la hiérarchie polynomiale (par exemple [32, 55]). En revanche, d'autres logiques étudiées dans le cadre des P2PIS, et ne nécessitant pas de recourir à des contraintes topologiques 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 PDMS de [63, 64] est ExpTime-complète pour SAT [48].

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 [8, 31]. Notons que l'inférence fondamentale de calcul de conséquences est utilisée en IA dans plusieurs tâches composites, telles que le raisonnement de sens commun 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 millier de pairs [7]. En ce qui concerne les BD, les PDMS SomeOWL [8] et SomeRDFS [9] 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 (propositionnalisable) 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 de RDFS (la recommandation du W3C pour les ontologies simples).

Contributions de cette thèse

Les contributions de cette thèse sont liés au raisonnement décentralisé dans les P2PIS propositionnels et à ses applications au Web Sémantique.

La première contribution est le premier algorithme décentralisé de calcul de conséquences par déduction linéaire dans les P2PIS propositionnels : DeCA$_{LR}$. Cette contribution inclus la première étude de complexité (communicationnelle, en espace et en temps) pour un algorithme totalement décentralisé de déduction, les algorithmes de la littérature n'ayant été jusqu'à présent évalués qu'expérimentalement.

La seconde contribution est la première étude de la notion d'extension conservative dans un cadre décentralisé. Notamment, nous exhibons son lien théorique avec la déduction linéraire décentralisée, et sa possible mise-en-œuvre avec DeCA$_{LR}$. Dans les P2PIS, cette notion est 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, 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 base de connaissance d'un pair donné par un P2PIS, de sorte à pouvoir l'empêcher.

La troisième contribution est un modèle distribué de données et les algorithmes associés de gestion de données afin de mettre en œuvre des PDMS pour le Web Sémantique fondés sur la logique de description DL-LITE$_{\mathcal{R}}$. Nous nous intéressons en particulier aux problèmes de vérification de consistance et de répondre à des requêtes. Notre approche consiste à réduire ces problèmes en des raisonnements en LP. Ceci permet de déployer de manière simple des PDMS pour DL-LITE$_{\mathcal{R}}$ "au-dessus" des P2PIS propositionnels et utiliser l'algorithme DeCA$_{LR}$ comme un moteur de raisonnement au niveau propositionnel.

La quatrième et dernière contribution est un modèle de données DL-LITE$_{\mathcal{R}}$ (centralisé ou décentralisé) étendu par des vues – l'accès au données se fait au travers de requêtes prédéfinies appelées vues – et les algorithmes associés de gestion de données au travers de vues. De nouveau, nous nous intéressons aux problèmes de vérification de la consistance et de répondre à des requêtes. Pour le cas centralisé, notre approche consiste à combiner les algorithmes proposés par [25] pour la gestion centralisée de données dans DL-LITE$_{\mathcal{R}}$ et l'algorithme MiniCon conçu par [45] pour le calcul de réécritures (maximales) en termes de vues. Pour le cas décentralisé, nous combinons les algorithmes décentralisés de gestion de données proposés dans la troisième contribution et l'algorithme MiniCon. Ceci permet de déployer des PDMS DL-LITE$_{\mathcal{R}}$ avec des vues au dessus des P2PIS propositionnels et utiliser DeCA$_{LR}$ pour raisonner au niveau propositionnel.

Plan de la dissertation

La première partie de cette thèse traite des raisonnements standards et non standards en logique propositionnelle. Nous rappelons tout d'abord la logique propositionnelle dans le Chapitre 1. Le problème de calcul de conséquences basé sur la déduction linéaire (raisonnement standard) est défini dans le Chapitre 2. Enfin, la notion d'extension (non) conservative dans les P2PIS propositionnels (raisonnement non standard) est traitée dans le Chapitre 3.

La seconde partie de cette thèse concerne l'application des contributions de la première partie au systèmes de gestion de données pour le Web Sémantique. Nous rappelons tout d'abord le modèle centralisé de DL-LITE$_{\mathcal{R}}$ [25] avec les algorithmes associés de vérification de consistance et de répondre à des requêtes dans le Chapitre 4. Le modèle distribuée de DL-LITE$_{\mathcal{R}}$, son encodage en LP et les algorithmes décentralisés de vérification de consistance et de réponse à des requêtes sont proposés dans le Chapitre 5. Enfin, le modèle de données (centralisé ou distribué) étendu par des vues et les algorithmes (centralisés et décentralisés) de vérification de la consistance et de réponse aux requêtes au travers de vues sont proposés dans le Chapitre 6.

Première partie

Raisonnement décentralisé en logique propositionnelle [1]

1. Les contributions de cette partie ont été publiées dans le journal européen *AI Communications* en 2009 [1]. Des travaux préliminaires ont été publiés aux *journées francophones de la programmation par contraintes* JFPC [2] et aux *journées de bases de données avancées* BDA [3] en 2008.

Chapitre 1

Préliminaires : la logique propositionnelle

Dans cette thèse, nous nous intéressons principalement à la *logique propositionnelle* (LP), la composante la plus simple de la logique classique. Nous rappelons tout d'abord la *syntaxe et la sémantique* de cette logique. Nous abordons ensuite la notion fondamentale de *conséquence* qui permet de formaliser la notion de raisonnement en logique. Plus précisément, nous présentons le problème de *calcul de ces conséquences* qui est au centre de cette thèse.

1.1 Syntaxe et sémantique

Étant donnés un alphabet $\mathcal{A}$ de variables propositionnelles et les constantes $vrai$ et $faux$, les *formules* de LP en termes de $\mathcal{A}$ sont définies par :

- $vrai$ et $faux$ sont des formules,
- $V \in \mathcal{A}$ est une formule,
- $\neg f$, où f est une formule, est une formule, et
- $(f_1 \vee f_2)$ est une formule.

De plus, $(f_1 \wedge f_2)$, $(f_1 \Rightarrow f_2)$ et $(f_1 \Leftrightarrow f_2)$, où f_1 et f_2 sont des formules, sont des formules définies respectivement par $\neg(\neg f_1 \vee \neg f_2)$, $(\neg f_1 \vee f_2)$ et $((f_1 \Rightarrow f_2) \wedge (f_2 \Rightarrow f_1))$.

La sémantique des formules de LP est fondée sur la notion d'*interprétation*. Une interprétation I assigne chaque variable V d'un alphabet $\mathcal{A}$ à $vrai$ ou $faux$. La notion de *satisfaction d'une formule* f par une interprétation I, notée $I \models f$, est définie par :

- $I \models vrai$ et $I \not\models faux$,
- $I \models V \in \mathcal{A}$ ssi $I(V) = vrai$,
- $I \models \neg f$ ssi $I \not\models f$, et
- $I \models (f_1 \vee f_2)$ ssi $I \models f_1$ ou $I \models f_2$.

Un *modèle d'une formule* est une interprétation qui satisfait cette formule et une formule est *satisfiable* si et seulement si elle a un modèle. Une *tautologie* est une formule qui est vraie dans toute interprétation (c-à-d. dont toute interprétation est un modèle).

Les théories propositionnelles. Une *théorie* est un ensemble fini de formules. De fait, la notion de théorie est purement syntaxique car une théorie constituée des formules $f_1, \ldots, f_n$ est sémantiquement équivalente à la formule $f_1 \wedge \cdots \wedge f_n$. Par conséquent, les notions de modèle et de satisfiablité d'une théorie découlent directement de celles définies pour une formule.

Les *théories clausales* jouent un rôle important en LP. Une théorie clausale est constituée uniquement de clauses, c'est-à-dire de disjonctions de littéraux, où un *littéral* est une variable propositionnelle ou sa négation. Un intérêt des *théories clausales* est de pouvoir représenter *n'importe quelles* connaissances propositionnelles dans une forme normale – un ensemble ou une conjonction de clauses – connue sous le nom de forme normale conjonctive (FNC). En effet, il est notoire que toute formule ou théorie peut être compilée en une FNC équivalente, bien que le résultat de la compilation puisse être – dans le pire des cas – exponentiellement plus grand que l'entrée. Un autre intérêt est que cette forme normale est commode pour développer des algorithmes de raisonnement, comme en témoignent les nombreux algorithmes de la littérarure utilisant des théories clausales, par exemple les algorithmes de preuves de théorèmes de [23] ou les algorithmes de calcul de conséquences de [55].

1.2 Le calcul de conséquences

Le *calcul de conséquences* consiste à dériver certains théorèmes implicites – les conséquences – d'une théorie. Ces théorèmes peuvent être, par exemple, ceux en termes d'un langage donné ou encore ceux résultant de l'ajout de nouvelles informations dans la théorie. Le calcul de conséquences joue un rôle important en IA car de nombreuses tâches de raisonnement peuvent s'y ramener : le raisonnement de sens commun, le diagnostic à base de modèles, la compilation de bases de connaissances, etc. Le lecteur intéressé par les fondements du calcul de conséquences pourra se référer à [55] qui offre un panorama des principaux résultats et applications en logique propositionnelle, ainsi qu'à [56, 49, 50] pour ce qui concerne la logique du premier ordre, à [18] pour les logiques de description et enfin à [19, 20] pour les logiques modales.

La notion de conséquence s'énonce formellement comme suit.

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

- *f est une* conséquence *de g, noté par $g \models f$, ssi tout modèle de g est un modèle de f,*
- *f est* équivalente *à g, noté par $f \equiv g$, ssi $f \models g$ et $g \models f$, et*
- *f est une* conséquence *de $\mathcal{T}$, noté par $\mathcal{T} \models f$, ssi tout modèle de $\mathcal{T}$ est un modèle de f.*

Lorsque la notion de conséquence est utilisée pour caractériser le contenu d'une théorie, on parle souvent d'*impliqués* de cette théorie pour ses conséquences clausales, et d'impliqués *premiers* de cette théorie pour ses conséquences clausales les plus fortes. De plus, on parle d'impliqué (premier) *propre* à une formule lorsque l'existence de cet impliqué découle de la présence de cette formule.

Définition 2 (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 f est une* conséquence *de $\mathcal{T}$.*
- *f est un* impliqué premier de $\mathcal{T}$ *ssi f est un impliqué de $\mathcal{T}$ et pour toute clause f' exprimée en termes de $\mathcal{A}$, si f' est un impliqué de $\mathcal{T}$ et $f' \models f$ alors $f \models f'$ (c-à-d. $f' \equiv f$).*
- *f est un* impliqué (premier) propre d'une formule q dans $\mathcal{T} \cup \{q\}$ *ssi f est un impliqué (premier) de $\mathcal{T} \cup \{q\}$, sans être un impliqué de $\mathcal{T} \backslash \{q\}$.*

Dans la suite, nous notons $\mathbf{PI}(\mathcal{T})$ la théorie clausale formée par tous les impliqués premiers de $\mathcal{T}$. La Propriété 1 formalise des résultats importants sur les liens existants entre $\mathcal{T}$ et $\mathbf{PI}(\mathcal{T})$ (*cf.* [55] par exemple). $\mathbf{PI}(\mathcal{T})$ fournit la caractérisation la *plus petite* (en nombre de clauses),

exacte et *explicite* de *toutes* les connaissances intensionnelles et extensionnelles modélisées par $\mathcal{T}$. De plus, lorsque $\mathcal{T}$ est une théorie clausale, cette caractérisation peut être exponentielle dans la taille (nombre de clauses) de $\mathcal{T}$.

Propriété 1 *Soit $\mathcal{T}$ une théorie en LP constituée de formules en termes d'un alphabet propositionnel $\mathcal{A}$.*

- $\mathbf{PI}(\mathcal{T})$ *existe toujours.*
- $\mathbf{PI}(\mathcal{T}) \equiv \mathcal{T}$.
- $\#\mathbf{PI}(\mathcal{T}) \leq 2^{\#\mathcal{T}}$ *quand $\mathcal{T}$ est une théorie clausale.*

Exemple. 1 (Impliqués premiers d'une théorie clausale)

$\neg FV \vee AV \vee PAS \vee KW$
$\neg F \vee \neg Cal$
$\neg AV \vee F$
$\neg AV \vee Cal$
$\neg FR \vee CAS \vee CER$
$\neg FR \vee Pol$

FIGURE 1.1 – Une théorie clausale $\mathcal{T}$ sur les *fruits*.

La Figure 1.1 illustre une théorie $\mathcal{T}$ constituée de six clauses. La première établit le fait que les fruits verts (FV) sont les avocats (AV), les pastèques (PAS), ou les kiwis (KW). La deuxième établit le fait que les fruits (F) ne contiennent pas beaucoup de calories (Cal). La troisième et la quatrième clauses établissent le fait que les avocats sont des fruits qui contiennent beaucoup de calories. Enfin, les deux dernières clauses précisent que les cerises (CER) et les cassis (CAS) sont des fruits rouges (FR) et que les fruits rouges sont riches en polyphénol (Pol).

L'ensemble des *impliqués* de $\mathcal{T}$ est $\{\neg FV \vee AV \vee PAS \vee KW, \neg F \vee \neg Cal, \neg AV \vee F, \neg AV \vee Cal, \neg FR \vee CAS \vee CER, \neg FR \vee Pol, \neg FV \vee F \vee PAS \vee KW, \neg FV \vee Cal \vee PAS \vee KW, \neg FV \vee \neg F \vee PAS \vee KW, \neg FV \vee \neg Cal \vee PAS \vee KW, \neg FV \vee PAS \vee KW, \neg AV \vee \neg Cal, \neg AV \vee \neg F, \neg AV\}$.

L'ensemble des *impliqués premiers* de $\mathcal{T}$ est $PI(\mathcal{T}) = \{\neg FR \vee CAS \vee CER, \neg FR \vee Pol, \neg AV, \neg F \vee \neg Cal, \neg FV \vee PAS \vee KW\}$. Il explicite toutes les connaissances de $\mathcal{T}$: les fruits rouges sont les cerises ou les cassis, les fuits rouges sont riches en polyphénol, il n'y a pas d'avocat, les fruits ne contiennent pas beaucoup de calories et les fruits verts sont les pastèques ou les kiwis. ■

1.2.1 La résolution - La déduction - La réfutation

La *résolution* [61] est un mécanisme clé pour trouver des impliqués d'une théorie propositionnelle clausale. La règle de résolution est la suivante :

$$\frac{l \vee l_1 \vee \cdots \vee l_m \quad \bar{l} \vee l'_1 \vee \cdots \vee l'_n}{l_1 \vee \cdots \vee l_m \vee l'_1 \vee \cdots \vee l'_n}.$$

Elle s'applique aux deux clauses $l \vee l_1 \vee \cdots \vee l_m$ et $\bar{l} \vee l'_1 \vee \cdots \vee l'_n$, où la première contient le complément d'un littéral de la deuxième : l. En effectuant une résolution sur l, cette règle produit le *résolvant* $l_1 \vee \cdots \vee l_m \vee l'_1 \vee \cdots \vee l'_n$ qui est un impliqué – donc une conséquence – de $l \vee l_1 \vee \cdots \vee l_m$ et $\bar{l} \vee l'_1 \vee \cdots \vee l'_n$. Par convention, un résolvant est supposé *réduit* : c'est soit la clause vide (équivalente à $faux$) notée $\square$, soit $vrai$ dans le cas d'une tautologie, et sinon une clause sans redondance de littéraux (les littéraux redondants sont supprimés).

La *déduction* est une séquence d'applications de la règle de résolution suivant une stratégie, dite *stratégie de résolution*. Les stratégies de résolution sont correctes pour le calcul de conséquences dans une théorie propositionnelle clausale, certaines d'entres elles étant aussi complètes. Une stratégie de résolution est correcte si elle ne dérive que des impliqués de cette théorie et complète si elle peut dériver tout impliqué premier de la théorie.

Dans cette thèse, nous considérons la déduction et la déduction linéaire qui sont correctes et complètes pour le calcul de conséquences (*cf.* [56, 49, 50, 55] par exemple).

La déduction n'impose pas de contraintes sur l'application de la règle de résolution : à chaque étape, la règle peut être appliquée sur n'importe quelle paire de clauses choisies parmi les clauses de la théorie en plus des résolvants déjà produits.

En revanche, la déduction linéaire impose qu'à chaque étape de la déduction la règle de résolution s'applique à une *center clause* et une *side clause*. Pour la première étape de déduction, la *center clause* est une clause désignée explicitement appelée *top clause* et la *side clause* est choisie parmi les clauses de la théorie. Pour toute étape de déduction ultérieure, la *center clause* est le résolvant de l'étape précédente et la *side clause* est choisie parmi les clauses de la théorie et les résolvants déjà obtenus. Par conséquent, la déduction linéaire est souvent représentée par un arbre dégénéré comme l'illustre la Figure 1.2.

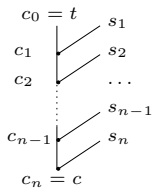

FIGURE 1.2 – Une déduction linéaire à n-étapes d'une clause c dans une théorie $\mathcal{T}$, ayant t comme *top clause*. c_i, avec $0 \leq i \leq n$, est une *center clause*. s_i, avec $1 \leq i \leq n$, est une *side clause* telle que $s_i \in \mathcal{T} \cup \{c_0, \ldots, c_{i-1}\}$. Une *center clause* c_i, avec $1 \leq i \leq n$, est le résolvant produit en appliquant la règle de résolution à la *center clause* c_{i-1} et la *side clause* s_i.

Désormais, on notera par $\mathcal{T} \vdash_R c$ le fait qu'il existe une *déduction* de c dans la théorie clausale $\mathcal{T}$ et par $\mathcal{T} \vdash^t_{LR} c$ le fait qu'il existe une *déduction linéaire* de c, ayant t comme *top clause*, dans la théorie clausale $\mathcal{T}$.

Exemple. 2 (Déduction/Déduction linéaire)
Considérons la théorie clausale $\mathcal{T}$ de la Figure 1.1, $\mathcal{T} = \{\neg FV \vee AV \vee PAS \vee KW, \neg F \vee \neg Cal, \neg AV \vee F, \neg AV \vee Cal, \neg FR \vee CAS \vee CER, \neg FR \vee Pol\}$.

- Une déduction possible de $\neg FV \vee PAS \vee KW$ consiste à appliquer la règle de résolution d'une part à $\neg FV \vee AV \vee PAS \vee KW$ et $\neg AV \vee F$, ce qui produit $\neg FV \vee F \vee PAS \vee KW$ $(*)$, et d'autre part à $\neg F \vee \neg Cal$ et $\neg AV \vee Cal$ ce qui produit $\neg F \vee \neg AV$ $(**)$. Une résolution entre $(*)$ et $(**)$ produit $\neg FV \vee \neg AV \vee PAS \vee KW$ $(***)$ et une dernière résolution entre $(***)$ et $\neg FV \vee AV \vee PAS \vee KW$ produit $\neg FV \vee PAS \vee KW$. Cette déduction est illustrée dans la Figure 1.3.
- Une déduction linéaire possible de $\neg FV \vee PAS \vee KW$ consiste à choisir parmi les clauses de $\mathcal{T}$ $c_0 = \neg AV \vee F$ comme *top clause* et $s_1 = \neg F \vee \neg Cal$ comme *side clause*. L'application de la règle de résolution à c_0 et s_1 produit $c_1 = \neg AV \vee \neg Cal$. La deuxième résolution

s'effectue entre c_1 et $s_2 = \neg AV \vee Cal$ et produit $c_2 = \neg AV$. Une dernière résolution entre c_2 et $s_3 = \neg FV \vee AV \vee PAS \vee KW$ produit $c_3 = c = \neg FV \vee PAS \vee KW$. Cette déduction linéaire est illustrée dans la Figure 1.4.

■

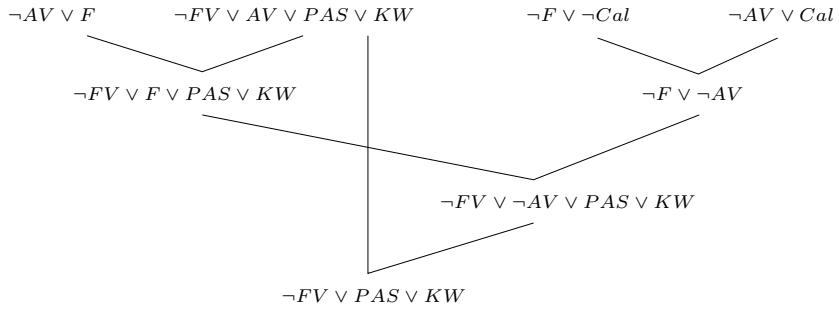

FIGURE 1.3 – Une déduction de $\neg FV \vee PAS \vee KW$ dans $\mathcal{T}$.

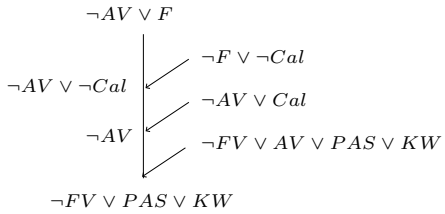

FIGURE 1.4 – Une déduction linéaire à 3 étapes de $\neg FV \vee PAS \vee KW$ dans $\mathcal{T}$, ayant $\neg AV \vee F$ comme *top clause*.

Nous rassemblons, par la Propriété 2, les principaux résultats sur la déduction (par exemple [56, 49, 50, 55]). Elle établit que – et montre comment – la déduction et la déduction linéaire sont correctes et complètes pour le calcul de conséquence (premier et deuxième points). Elle établit aussi que – et montre comment – la déduction linéaire peut être utilisée pour trouver des impliqués de la théorie et la *top clause*, peu importe si cette *top clause* appartient à la théorie, y compris *tous* les impliqués premiers qui découlent obligatoirement de cette *top clause* (troisième point) : les impliqués premiers propres de la top clause. Notons que le troisième point implique le deuxième.

Propriété 2 *Soit $\mathcal{T}$ une théorie propotionnelle clausale en termes d'un alphabet propositionnel $\mathcal{A}$. Soit r une clause en termes de $\mathcal{A}$.*

- $\mathbf{PI}(\mathcal{T}) \subseteq \{c \mid \mathcal{T} \vdash_R c\} \subseteq \{c \mid \mathcal{T} \models c\}$
- $\mathbf{PI}(\mathcal{T}) \subseteq \{c \mid \exists t \in \mathcal{T}, \mathcal{T} \vdash^t_{LR} c\} \subseteq \{c \mid \mathcal{T} \models c\}$
- $\{c \mid c \in \mathbf{PI}(\mathcal{T} \cup \{r\})$ *et* $\mathcal{T} \backslash \{r\} \not\models c\} \subseteq \{c \mid \mathcal{T} \vdash^r_{LR} c\} \subseteq \{c \mid \mathcal{T} \cup \{r\} \models c\}$

Enfin, la *réfutation* est une utilisation particulière de la déduction. Elle revient à tester si une clause c est une conséquence d'une théorie $\mathcal{T}$ en vérifiant si la négation de c et cette théorie sont insatisfiables : en effet $\mathcal{T} \models c$ ssi $\mathcal{T} \cup \{\neg c\} \models \square$. Par conséquent, nous avons $\mathcal{T} \vdash_R c$ ssi $\mathcal{T} \cup \{\neg c\} \vdash_R \square$ puisqu'un ensemble insatisfiable de clauses a un seul impliqué premier, $\square$, et que la déduction est complète pour le calcul de conséquences.

Exemple. 3 (Réfutation)
Considérons la théorie clausale $\mathcal{T}$ de la Figure 1.1, $\mathcal{T} = \{\neg FV \vee AV \vee PAS \vee KW, \neg F \vee \neg Cal, \neg AV \vee F, \neg AV \vee Cal, \neg FR \vee CAS \vee CER, \neg FR \vee Pol\}$.

Une déduction de $\neg FV \vee PAS \vee KW$ par réfutation consiste à (i) injecter $\neg(\neg FV \vee PAS \vee KW)$ dans $\mathcal{T}$, c'est-à-dire FV, $\neg PAS$ et $\neg KW$, puis (ii) à produire $\square$ par déduction dans la théorie résultante. Une telle réfutation est illustrée dans la Figure 1.5. ■

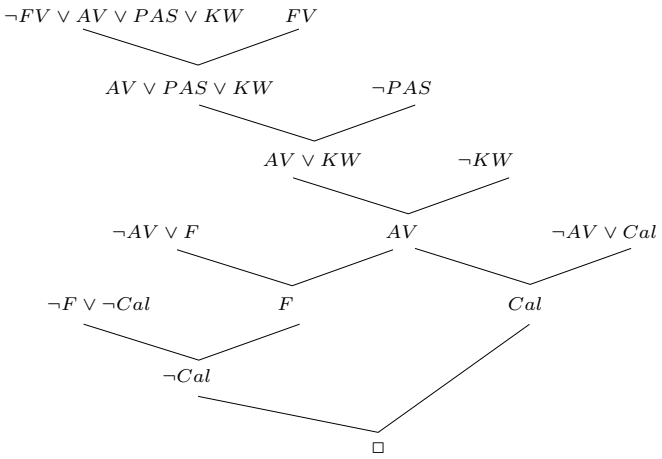

FIGURE 1.5 – Une réfutation de $\neg FV \vee PAS \vee KW$ dans $\mathcal{T}$, c-à-d. une déduction de $\square$ dans $\mathcal{T} \cup \{FV, \neg PAS, \neg KW\}$
.

Chapitre 2

Déduction *linéaire* décentralisée

Nous présentons dans ce chapitre la première contribution de cette thèse : un algorithme décentralisé de calcul de conséquences fondé sur la déduction linéaire en logique propositionnelle. Pour cela, nous nous plaçons dans le cadre distribué et complètement décentralisé des systèmes d'inférence pair-à-pair (P2PIS) propositionnels.

Le chapitre est organisé comme suit. Nous définissons tout d'abord les P2PIS propositionnels auxquels nous nous intéressons dans la Section 2.1. Nous abordons ensuite la déduction linéaire décentralisée pour ces systèmes dans la Section 2.2. Enfin dans la Section 2.3, nous présentons les principaux travaux de la littérature sur la déduction décentralisée en logique propositionnelle.

2.1 Les systèmes d'inférence pair-à-pair (P2PIS) propositionnels

Un système d'inférence pair-à-pair $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ est un ensemble de pairs, où l'indice i modélise l'*identifiant* du pair $\mathcal{P}_i$ dans $\mathcal{S}$ (par exemple son adresse IP). Un pair $\mathcal{P}_i$ gère des connaissances modélisées par une *théorie clausale* 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 (non tautologiques) 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 exemple 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 $Tgt(\mathcal{A}(\mathcal{P}_i))$. C'est le langage qu'utilise $\mathcal{P}_i$ pour répondre aux questions qui lui sont posées.
Nous supposons que la clause vide $\square$ appartient à la fois à $\mathcal{L}(\mathcal{P}_i)$ et à $Tgt(\mathcal{L}(\mathcal{P}_i))$.

Un mapping de $\mathcal{P}_i$ est une clause (non tautologique) sans redondance de littéral et dont les littéraux utilisent des variables de $\mathcal{P}_i$ mais aussi des variables appartenant à d'autres pairs du système. Un mapping avec n littéraux peut alors impliquer jusqu'à n pairs. Dans ce travail, nous adoptons le principe de stocker un mapping dans tous les pairs qu'il implique. Par conséquent, les mappings dans lesquels $\mathcal{P}_i$ est impliqué sont stockés localement dans $\mathcal{M}(\mathcal{P}_i)$.

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. L'alphabet de $\mathcal{S}$, $\mathcal{A}(\mathcal{S})$, est l'union des alphabets de ses pairs. L'alphabet cible de $\mathcal{S}$, $Tgt(\mathcal{A}(\mathcal{S}))$, est l'union des alphabets cibles de ses pairs. Son langage $\mathcal{L}(\mathcal{S})$ (respectivement son langage cible $Tgt(\mathcal{L}(\mathcal{S}))$) est l'ensemble fini de clauses (non tautologiques) sans répétition de littéral exprimées en termes de $\mathcal{A}(\mathcal{S})$ (respectivement de $Tgt(\mathcal{A}(\mathcal{S}))$).
Notons que d'un point de vue logique, un P2PIS $\mathcal{S}$ est la théorie propositionnelle clausale – distribuée – $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ en termes de $\mathcal{A}(\mathcal{S})$.

La sémantique d'un pair $\mathcal{P}_i$ et d'un P2PIS $\mathcal{S}$ découle – bien évidemment – de celle des théories $\mathcal{T}(\mathcal{P}_i) \cup \mathcal{M}(\mathcal{P}_i)$ et $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$. Les notions d'interprétation, de modèle, et de satisfiabilité d'une théorie de LP s'appliquent alors directement à un pair ou à un P2PIS.

Exemple. 4 (Service distribué de cours particuliers)
La Figure 2.1 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.

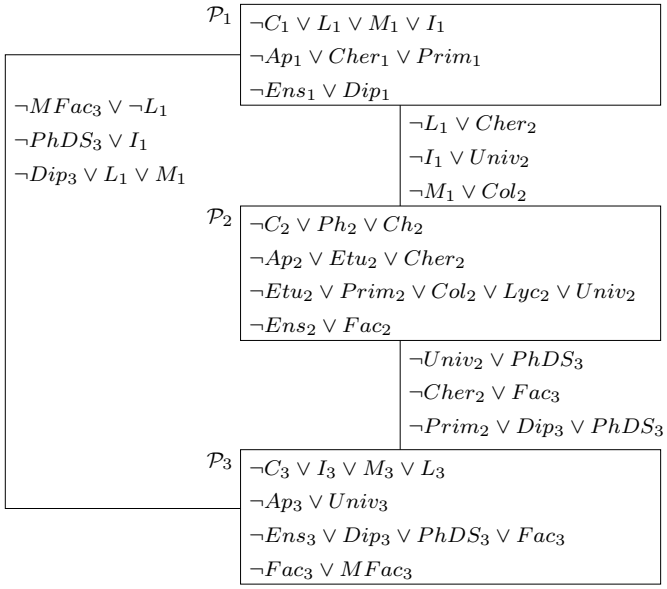

FIGURE 2.1 – Le P2PIS $\mathcal{S}$.

Chaque pair fait de la publicité pour un centre de cours particuliers en décrivant ses cours, ses apprentis, ses enseignants et ses collaborations avec les autres centres du domaine.

$\mathcal{P}_1$ établit que le premier centre offre des cours (C_1) de langues (L_1), de maths (M_1), ou d'informatique (I_1). Les cours sont offerts aux apprentis du centre (Ap_1) qui sont des chercheurs ($Cher_1$) ou des étudiants en classe primaire ($Prim_1$) et sont assurés par des enseignants (Ens_1) diplômés (Dip_1).

$\mathcal{P}_2$ établit que le deuxième centre offre des cours (C_2) de physique (Ph_2) ou de chimie (Ch_2) à ses apprentis (Ap_2) qui sont des chercheurs ($Cher_2$) ou des étudiants (Etu_2) en école primaire

($Prim_2$), au collège (Col_2), au lycée (Lyc_2), ou encore à l'université ($Univ_2$). De plus, les enseignants (Ens_2) sont des enseignants en faculté (Fac_2).

Enfin, $\mathcal{P}_3$ établit que le troisième centre offre des cours (C_3) de langues (L_3), de maths (M_3), ou d'informatique (I_3), à des apprentis (Ap_3) universitaires ($Univ_3$), et embauche des enseignants (Ens_3) diplômés (Dip_3), en thèse ($PhDS_3$), enseignants en faculté (Fac_3) et spécialisés en mathématiques ($MFac_3$).

Les trois centres collaborent ensemble selon la politique modélisée par les mappings :

- Le premier centre fournit des cours aux étudiants du deuxième centre : il fournit des cours de langues (L_1) aux chercheurs ($Cher_2$), des cours d'informatique (I_1) aux universitaires ($Univ_2$), et des cours de maths (M_1) aux collégiens (Col_2).
- Le deuxième centre sollicite des enseignants du troisième centre : les universitaires ($Univ_2$) sont enseignés par des étudiants en thèse ($PhDS_3$), les chercheurs ($Cher_2$) par des enseignants en faculté (Fac_3), et les étudiants en primaire ($Prim_2$) par des diplômés (Dip_3) ou des thésards ($PhDS_3$).
- Enfin, les enseignants du troisième centre enseignent (ou pas) certains cours au apprentis du premier centre : les enseignants en faculté spécialisés en maths ($MFac_3$) ne donnent pas de cours de langues (L_1), les thésards ($PhDS_3$) donnent des cours d'informatique (I_1), et les diplômés (Dip_3) donnent des cours de langues (L_1) ou de maths (M_1). ■

2.2 Déduction linéaire décentralisée

Intéressons-nous maintenant au problème de déduction linéaire dans les P2PIS propositionnels. Nous commençons par définir ce problème. Nous présentons ensuite les fondements théoriques permettant d'aboutir à notre algorithme de déduction linéaire décentralisé DeCA$_{LR}$, pour lequel nous établissons ses propriétés (correction, complétude et terminaison) et ses complexités (communication, espace et temps).

2.2.1 Définition du problème

Notre problème consiste à faire de la déduction linéaire dans une théorie distribuée sur les pairs d'un P2PIS propositionnel. D'un point de vue logique, ce problème est identique à la déduction linéaire dans une théorie centralisée. La difficulté est ici algorithmique car fournir une solution à ce problème impose un algorithme de déduction linéaire totalement décentralisé (aucun acteur d'un P2PIS n'ayant une vision globale du système).

Définition 3 (Déduction linéaire décentralisée) *Étant donné un P2PIS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$, une* déduction linéaire décentralisée *consiste à calculer des conséquences – en suivant la stratégie linéaire d'application de la règle de résolution – à partir de la théorie distribuée $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ et d'une top clause q exprimée dans le langage d'un pair $\mathcal{P}$ de $\mathcal{S}$. En particulier, une telle déduction notée $\vdash^q_{LR}$ doit terminer, être correcte pour le calcul d'impliqués et complète pour le calcul d'impliqués premiers (propres).*

2.2.2 Fondements de l'algorithme DeCA$_{LR}$

Pour produire des déductions linéaires décentralisées dans un P2PIS propositionnel, nous adoptons le principe suivant. Étant donné une *top clause* fournie à un pair – cette top clause

étant dans son langage –, ce pair raisonne dans un premier temps *localement* : il produit tous les impliqués dérivables linéairement à partir de la *top clause* fournie et de ses connaissances propres (sa théorie clausale et ses mappings avec les autres pairs). Dans un second temps, à partir des résultats de déductions linéaires décentralisées adéquates déclenchées sur des pairs distants (qui procéderont à leur tour sur ce même principe), le pair construit *des* impliqués du P2PIS et de la top clause fournie, y compris *tous* les impliqués premiers découlant nécessairement de cette top clause.

La Proposition 1 fournit les fondements nécessaires pour implémenter le principe de déduction linéaire décentralisée décrit ci-dessus. Elle utilise l'opérateur de distribution clausale $\otimes$ qui est associatif et commutatif. Il s'applique à deux ensembles de clauses et produit un ensemble de clauses *réduites* tel que $\emptyset \otimes \mathcal{C} = \emptyset$, $\{\Box\} \otimes \mathcal{C} = \mathcal{C}$ et $\{c^1, \ldots, c^m\} \otimes \{c_1, \ldots, c_n\} = \{c^1 \vee c_1, \ldots, c^1 \vee c_n, \ldots, c^m \vee c_1, \ldots, c^m \vee c_n\}$. La Proposition 1 établit en effet que *des* impliqués du P2PIS et de la top clause, y compris *tous* les impliqués premiers qui découlent nécessairement de la top clause, sont produits en *décomposant* chaque impliqué dérivé localement ($l_k^1 \vee \cdots \vee l_k^n \vee l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ dans la Proposition 1) en deux parties, une partie locale ($l_k^1 \vee \cdots \vee l_k^n$ dans la Proposition 1) et une partie non locale ($l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ dans la Proposition 1), puis en *combinant* par $\otimes$ sa partie locale avec les impliqués dérivés linéairement dans le P2PIS à partir de sa partie non locale.

Proposition 1 *Soient $\mathcal{S}$ un P2PIS, $\mathcal{P}_k$ un pair de $\mathcal{S}$ et q_k une clause de $\mathcal{L}(\mathcal{P}_k)$.*

1. *Si $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \vdash^{q_k}_{LR} l_k^1 \vee \cdots \vee l_k^n \vee l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ et $c \in \{l_k^1 \vee \cdots \vee l_k^n\} \otimes \{c' \mid \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \vdash^{l^{n+1}_{\alpha_{n+1}} \vee \cdots \vee l^p_{\alpha_p}}_{LR} c'\}$ alors $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \models c$.*

2. *Si $c = l_k^1 \vee \cdots \vee l_k^m \vee l^{m+1}_{\beta_{m+1} \neq k} \vee \cdots \vee l^q_{\beta_q \neq k} \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\})$ et $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \backslash \{q_k\} \not\models c$ alors $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \vdash^{q_k}_{LR} l_k^1 \vee \cdots \vee l_k^{n \leq m} \vee l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ et $c \in \{l_k^1 \vee \cdots \vee l_k^n\} \otimes \{c' \mid \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \vdash^{l^{n+1}_{\alpha_{n+1}} \vee \cdots \vee l^p_{\alpha_p}}_{LR} c'\}$.*

Preuve. Considérons le premier point. Si $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \vdash^{q_k}_{LR} l_k^1 \vee \cdots \vee l_k^n \vee l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$, alors $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\} \models l_k^1 \vee \cdots \vee l_k^n \vee l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ (correction de la déduction linéaire pour le calcul de conséquences). Puisque $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \subseteq \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$, nous avons $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \models l_k^1 \vee \cdots \vee l_k^n \vee l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ (monotonie de $\models$ pour la logique propositionnelle). Soit I un modèle de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\}$.
Si $I \models l_k^1 \vee \cdots \vee l_k^n$ alors $I \models c \in \{l_k^1 \vee \cdots \vee l_k^n\} \otimes \{c' \mid \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \vdash^{l^{n+1}_{\alpha_{n+1}} \vee \cdots \vee l^p_{\alpha_p}}_{LR} c'\}$.
Sinon, $I \models l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}$ et par conséquent $I \in \{J \mid J \models \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \cup \{l^{n+1}_{\alpha_{n+1} \neq k} \vee \cdots \vee l^p_{\alpha_p \neq k}\}\}$. Puisque $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \vdash^{l^{n+1}_{\alpha_{n+1}} \vee \cdots \vee l^p_{\alpha_p}}_{LR} c'$, nous avons $I \models c'$ (correction de la déduction linéaire pour le calcul de conséquences), et donc $I \models c \in \{l_k^1 \vee \cdots \vee l_k^n\} \otimes \{c' \mid \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \vdash^{l^{n+1}_{\alpha_{n+1}} \vee \cdots \vee l^p_{\alpha_p}}_{LR} c'\}$.

Considérons maintenant le second point. Puisque $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\})$ et $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \backslash \{q_k\} \not\models c$ alors $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^{q_k}_{LR} c$ (complétude de la déduction linéaire pour le calcul des impliqués premiers propres). Observons tout d'abord qu'il est toujours possible de construire une dérivation équivalente à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^{q_k}_{LR} c$ telle qu'aucune side clause n'est une center clause produite précédemment. En effet, si tel est le cas, une telle side clause peut être substituée par

une dérivation utilisant les clauses ayant créée la center clause. Nous considérons donc dans la suite, sans perte de généralité, que $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ ne réutilise aucune center clause comme side clause.

Si toute center clause du préfixe de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ local à $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\}$, c-à-d. n'utilisant que des clauses de $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\}$, est de la forme $l_k^1 \vee \cdots \vee l_k^{n>m} \vee l_{\alpha_{n+1} \neq k}^{n+1} \vee \cdots \vee l_{\alpha_p \neq k}^{p}$, alors cette dérivation effectue ultérieurement des applications de la règle de résolution sur $l_k^{m+1}, \ldots, l_k^n$ grâce à des side clauses $\bar{l}_k^{m+1} \vee \cdots, \ldots, \bar{l}_k^n \vee \cdots$ appartenant par définition à $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\}$ (elles contiennent des littéraux de $\mathcal{P}_k$). Montrons alors que nous pouvons construire une autre déduction de c à partir de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$, dans laquelle les applications de la règle de résolution sur $l_k^{m+1}, \ldots, l_k^n$ grâce aux clauses $\bar{l}_k^{m+1} \vee \cdots, \ldots, \bar{l}_k^n \vee \cdots$ se feront dans le préfixe de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ local à $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\}$.

Supposons que la première résolution sur $l_k^i \in \{l_k^{m+1}, \ldots, l_k^n\}$ s'effectue lors de la $j^{\text{ième}}$ étape de résolution de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ (voir Figure 2.2).

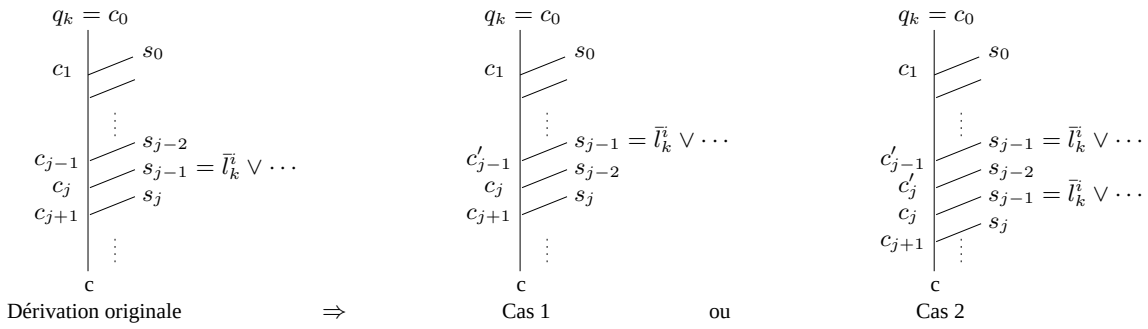

FIGURE 2.2 – Exemple sur la manière par laquelle on peut toujours construire une autre déduction de c à partir de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$, dans laquelle l' application de la règle de résolution sur l_k^i grâce à la clause $\bar{l}_k^i$ se fait dans le préfixe de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ local à $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\}$. Dans la dérivation initiale, la première résolution sur l_k^i s'effectue lors de la $j^{\text{ième}}$ étape de résolution de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$. Dans le Cas 1, la résolution avec s_{j-1} est permutée avec la résolution avec s_{j-2} et c_j est toujours obtenue à la $j^{\text{ième}}$ étape. Dans le Cas 2, la résolution avec s_{j-1} est permutée avec la résolution avec s_{j-2} et une nouvelle clause c'_j est obtenue à la $j^{\text{ième}}$ étape. Dans ce cas, une deuxième résolution avec s_{j-1} est ajoutée juste après celle avec s_{j-2} pour éliminer le littéral l_k^i avant de poursuivre la déduction.

Cas 1. Si la permutation de s_{j-1} avec s_{j-2} dans $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ permet toujours de dériver c_j alors nous avons une nouvelle dérivation linéaire de c qui diffère de la dérivation originale uniquement sur la permutation de s_{j-1} avec s_{j-2} et une nouvelle center clause c'_{j-1}. Notons que le changement de center clause c'_{j-1} n'a pas d'influence sur le reste de la dérivation car aucune center clause n'est réutilisée comme side clause.

Cas 2. Si la permutation de s_{j-1} avec s_{j-2} dans $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ ne permet plus de dériver la même center clause c_j à la j^{ime} étape de résolution, alors c'est que s_{j-2} contient l_k^i. Dans ce cas, il suffit de faire suivre la permutation de s_{j-1} avec s_{j-2} d'une nouvelle étape de résolution sur l_k^i entre s_{j-1} et la nouvelle center clause c'_j : la clause obtenue est alors la clause c_j de la dérivation originale. On obtient ainsi une nouvelle dérivation linéaire de c qui diffère de la dérivation originale uniquement sur la permutation de s_{j-1} avec s_{j-2}, l'introduction d'une nouvelle étape de résolution et des nouvelles center clauses c'_{j-1} et c'_j. Notons qu'ici aussi, le

changement des center clauses n'a pas d'influence sur le reste de la dérivation car aucune center clause n'est réutilisée comme side clause.

En appliquant autant de fois que nécessaire le résultat précédent, il est possible de construire une autre déduction de c à partir de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$, dans laquelle les applications de la règle de résolution sur $l_k^{m+1}, \ldots, l_k^n$ grâce aux clauses $\bar{l}_k^{m+1} \vee \cdots, \ldots, \bar{l}_k^n \vee \cdots$ se feront dans le préfixe de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ local à $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup \{q_k\}$. □

Il est important de noter que la Proposition 1 fournit seulement un début de solution pour mettre en œuvre notre principe de déduction linéaire décentralisée exposé ci-dessus. En effet, nous ne pouvons pas ré-appliquer directement la Proposition 1 à elle-même afin de calculer $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \vdash_{LR}^{l_{\alpha_{n+1}}^{n+1} \vee \cdots \vee l_{\alpha_p}^p} c'$, car $l_{\alpha_{n+1}}^{n+1} \vee \cdots \vee l_{\alpha_p}^p$ n'est pas, dans le cas général, dans le langage d'un *unique* pair. Nous décomposons alors ce calcul en plusieurs déductions linéaires décentralisées indépendantes dont les top clauses sont unitaires : des littéraux. Chaque top clause est alors dans le langage d'un unique pair et chacune des déductions linéaires décentralisées peut être effectuée – en suivant la Proposition 1 – par le pair distant auquel appartient la top clause unitaire.

La Proposition 2 établit comment on peut faire cela en montrant que des déductions linéaires indépendantes peuvent être combinées de sorte que leur combinaison est correcte et complète pour le calcul de conséquences : elle produit *des* impliqués (correction, premier point), y compris *tous* les impliqués premiers qui découlent nécessairement de cette top clause (complétude, deuxième point). On remarquera que la Proposition 2 montre aussi comment des déductions linéaires peuvent être *parallélisées*.

Proposition 2 *Soient $\mathcal{T}$ une théorie clausale de LP et $l_1 \vee \cdots \vee l_n$ une clause.*

1. *Si $c \in \mathbb{\odot}_{i=1}^n \{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$ alors $\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\} \models c$.*
2. *Si $c \in \mathbf{PI}(\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\})$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \not\models c$ alors $c \in \mathbb{\odot}_{i=1}^n \{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$.*

Preuve. Considérons le premier point. Il découle directement de la correction de la déduction linéaire pour le calcul de conséquences. En effet, pour tout $i \in [1..n]$ c_i est vrai dans tous les modèles de $\mathcal{T}$ où l_i est vrai. Pour tout modèle de $\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\}$, il existe $i \in [1..n]$ tel que l_i est vrai, donc où c_i est vrai et par conséquent où c est vrai.

Considérons maintenant le second point. Il découle de la Propriété de *distribution* de la caractérisation des impliqués premiers (Proposition 84 de [55]) et de la complétude de la déduction linéaire pour le calcul de conséquences (Propriété 2 du chapitre 1).

La Propriété de distribution établit que les impliqués premiers propres d'une clause donnée sont obtenus en substituant les littéraux de cette clause par leurs impliqués premiers propres : $\{c \mid c \in \mathbf{PI}(\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\})$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \not\models c\} \subseteq \{c_1 \vee \cdots \vee c_n \mid \forall i \in [1..n],\ c_i \in \mathbf{PI}(\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \cup \{l_i\})$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n, l_i\} \not\models c_i\}$.

La complétude de la déduction linéaire pour le calcul de conséquences assure alors : $\{c \mid c \in \mathbf{PI}(\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\})$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \not\models c\} \subseteq \{c_1 \vee \cdots \vee c_n \mid \forall i \in [1..n], \mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \vdash_{LR}^{l_i} c_i$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n, l_i\} \not\models c_i\}$. Par conséquent, $\{c \mid c \in \mathbf{PI}(\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\})$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \not\models c\} \subseteq \{c_1 \vee \cdots \vee c_n \mid \forall i \in [1..n], \mathcal{T} \vdash_{LR}^{l_i} c_i\}$.

Finalement, par définition de l'opérateur de distribution clausale $\mathbb{\odot}$, nous obtenons : $\{c \mid c \in \mathbf{PI}(\mathcal{T} \cup \{l_1 \vee \cdots \vee l_n\})$ et $\mathcal{T} \backslash \{l_1 \vee \cdots \vee l_n\} \not\models c\} \subseteq \mathbb{\odot}_{i=1}^n \{c_i \mid \mathcal{T} \vdash_{LR}^{l_i} c_i\}$. □

2.2.3 L'algorithme DeCA$_{LR}$

Maintenant que nous avons un principe de déduction linéaire décentralisée correct et complet pour le calcul de conséquences – grâce aux Propositions 1 et 2 –, voyons comment celui-ci est mis en œuvre dans l'algorithme DeCA$_{LR}$.

Description

DeCA$_{LR}$ (Algorithme 1) est un algorithme *récursif* décentralisé qui s'exécute sur chacun des pairs du P2PIS, un appel récursif signifiant qu'un pair en sollicite un autre pour décentraliser une déduction linéaire – grâce aux Propositions 1 et 2 –. Le point subtile est que DeCA$_{LR}$ s'exécutant sur le pair $\mathcal{P}_k$ est appelé DeCA$_{LR}{}^k$ et qu'ainsi $\mathcal{P}_k$ sollicite un pair $\mathcal{P}_j$ en déclenchant un appel récursif de DeCA$_{LR}{}^k$ à DeCA$_{LR}{}^j$.

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).

La première raison d'utiliser un historique vient de la Proposition 1. En effet, à chaque fois qu'un pair décompose un impliqué dérivé localement afin de décentraliser une dérivation, les dérivations linéaires résultantes doivent être produites en utilisant les clauses du P2PIS *et la top clause* à partir de laquelle l'impliqué décomposé a été obtenu : $\mathcal{T}(\mathcal{S})\cup\mathcal{M}(\mathcal{S})\cup\{\mathbf{q_k}\} \vdash_{LR}^{l_{\alpha_{n+1}}^{n+1}\vee\cdots\vee l_{\alpha_p}^{p}} c'$ dans la Proposition 1.

La seconde raison d'utiliser un historique est 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.

Exemple. 5 (Usage de l'historique) La Figure 2.3 illustre l'usage de l'historique sur un exemple d'une déduction linéaire décentralisée au sein du P2PIS $\mathcal{S}$ de la Figure 2.1 avec la top clause $Lyc_2 \vee Prim_2$.

Chaque noeud correspond à une dérivation linéaire locale à un pair, le nom de ce pair étiquetant le nœud. Cette dérivation locale utilise la top clause et l'historique fournie sur la flèche entrante. Les flèches sortantes indiquent que les Propositions 1 et 2 ont été appliquées sur le résultat de la dérivation locale et fournissent les top clauses et les historiques résultants. Chaque flèche possède deux étiquettes : (i) la top clause et l'historique fournis à un noeud et (ii) un impliqué calculé à partir de cette top clause et de cet historique. Remarquons que la clause vide ne peut pas être dérivée sans l'usage de l'historique (ici, sans L_1) dans le noeud en bas de la Figure et que l'historique a été utilisé pour la détection d'un cycle dans une branche de raisonnement (représentée par la flèche en pointillé). ■

Nous décrivons maintenant plus précisément le calcul effectué par DeCA$_{LR}$ (Algorithme 1).

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).

Puis, DeCA$_{LR}{}^k$ décompose les impliqués dérivés localement grâce aux Propositions 1 et 2 (lignes 9 et 10), de sorte à produire des nouveaux impliqués (premiers) du P2PIS et de la top clause en sollicitant des pairs distants – qui exécutent le même algorithme –.

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). Notons que nous présentons aussi les conditions de filtrage appropriées pour retourner les clauses

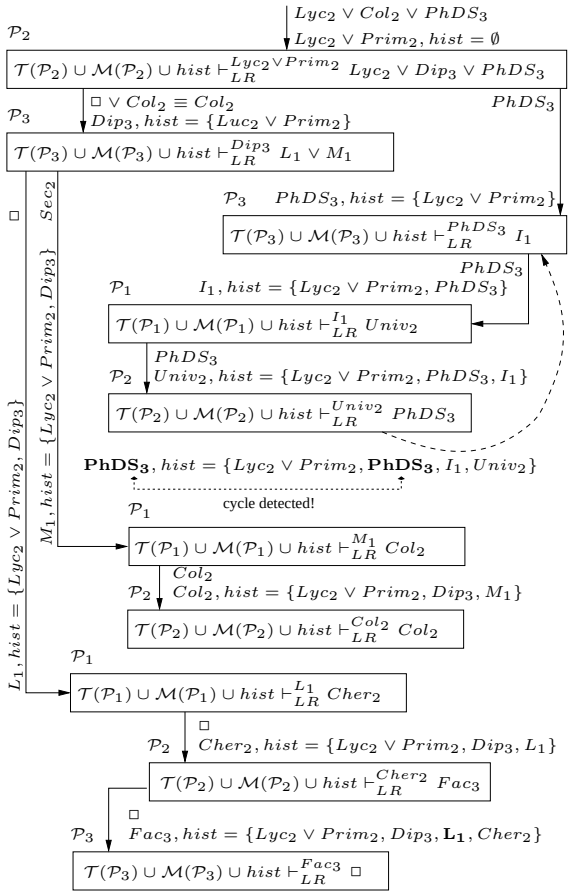

FIGURE 2.3 – Exemple d'une déduction linéaire décentralisée dans le P2PIS $\mathcal{S}$ de la Figure 2.1, ayant $Lyc_2 \vee Prim_2$ comme top clause.

dans le langage d'un pair donné $\mathcal{P}_{id}$ du P2PIS (lignes 8 et 13). Ceci sera utile dans le Chapitre 3, lorsque DECA$_{LR}$ sera appelé par l'algorithme CECA de test d'extension conservative.

Exemple. 6 (Exécution de DECA$_{LR}$**)** Nous montrons sur le P2PIS $\mathcal{S}$ de la Figure 2.1 comment DECA$_{LR}$ calcule des impliqués de $\mathcal{S}$ et la top clause $Lyc_2 \vee Prim_2$, y compris tous les impliqués premiers propres à cette top clause. Notons que la dérivation de la Figure 2.3 est *une* dérivation linéaire produite par DECA$_{LR}$ pour la top clause $Lyc_2 \vee Prim_2$.

Les déductions linéaires décentralisées sont initiées par DECA$_{LR}{}^2(Lyc_2 \vee Prim_2, \emptyset)$.

Cet appel commence par les dérivations linéaires locales et produit $CI = \{Lyc_2 \vee Prim_2, Lyc_2 \vee Dip_3 \vee PhDS_3\}$ à la ligne 4. Puisque $Lyc_2 \vee Dip_3 \vee PhDS_3$ contient des littéraux du pair distant $\mathcal{P}_3$, DECA$_{LR}{}^3(Dip_3, \{Lyc_2 \vee Prim_2\})$ et DECA$_{LR}{}^3(PhDS_3, \{Lyc_2 \vee Prim_2\})$ sont déclenchés et leurs résultats sont ensuite combinés avec $\{Lyc_2\}$ en utilisant $\curlyvee$, à la ligne 10, afin d'obtenir de nouveaux impliqués du P2PIS et $Lyc_2 \vee Prim_2$.

Dans notre cas, DECA$_{LR}{}^3(Dip_3, \{Lyc_2 \vee Prim_2\}) = \{Dip_3, M_1, Col_2\}$ et DECA$_{LR}{}^3(PhDS_3, \{Lyc_2 \vee Prim_2\}) = \{PhDS_3, I_1, Univ_2\}$ (voir ci-dessous), ainsi on obtient DECA$_{LR}{}^2(Lyc_2 \vee Prim_2, \emptyset) = \{Lyc_2 \vee Prim_2, Lyc_2 \vee Dip_3 \vee PhDS_3, Lyc_2 \vee Dip_3 \vee PhDS_3, Lyc_2 \vee Dip_3 \vee I_1, Lyc_2 \vee Dip_3 \vee Univ_2, Lyc_2 \vee M_1 \vee PhDS_3, Lyc_2 \vee M_1 \vee I_1, Lyc_2 \vee M_1 \vee Univ_2, Lyc_2 \vee Col_2 \vee PhDS_3, Lyc_2 \vee$

Algorithme 1: DECA$_{LR}$ sur le pair $\mathcal{P}_k$ du P2PIS $\mathcal{S}$. [Le texte entre crochets ajoute des *optimisations* quand DECA$_{LR}$ sera appelé par CECA (Algorithme 2 du Chapitre 3).]
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 d'un pair id]
Sortie: un ensemble de conséquences 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 est déjà traité)
(2) **return** $\emptyset$ % plus de nouveaux impliqués (cycle)
(3) **else**
(4) $CI \leftarrow \{c \mid \mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup hist \vdash_{LR}^{q_k} c\}$
(5) **if** $\square \in CI$
(6) **return** $\{\square\}$ % plus de nouveaux impliqués premiers ($\square$ est premier)
(7) **else**
(8) [**if** $k \neq id$ supprimer de CI toute clause ayant au moins un littéral $l \in \mathcal{L}(\mathcal{P}_k)$]
(9) **foreach** clause $l_k^1 \vee \cdots \vee l_k^q \vee l_{\alpha_{q+1}}^{q+1} \vee \cdots \vee l_{\alpha_n}^n \in CI$[% when $k \neq id$, $l_k^1 \vee \cdots \vee l_k^q = \square$ because of line 8]
(10) $CI \leftarrow CI \cup \{l_k^1 \vee \cdots \vee l_k^q\} \otimes \otimes_{i=q+1}^n$DECA$_{LR}{}^{\alpha_i}(l_{\alpha_i}^i, hist \cup \{q_k\})$
(11) **if** $\square \in CI$
(12) **return** $\{\square\}$ % $\square$ est le seul impliqué premier
(13) [supprimer de CI toute clause qui n'est pas dans $\mathcal{L}(\mathcal{P}_{id})$]
(14) **return** CI

$Col_2 \vee I_1, Lyc_2 \vee Col_2 \vee Univ_2\}$.

Nous détaillons tout d'abord DECA$_{LR}{}^3(PhDS_3, \{Lyc_2 \vee Prim_2\})$ qui calcule des impliqués du P2PIS avec $PhDS_3$ et $Lyc_2 \vee Prim_2$, y compris tous les impliqués premiers propres à $PhDS_3$.

Il commence par les déductions linéaires locales, ce qui donne à la ligne 4 $CI = \{PhDS_3, I_1\}$. Puisque I_1 est un littéral dans le langage du pair distant $\mathcal{P}_1$, DECA$_{LR}{}^1(I_1, \{Lyc_2 \vee Prim_2, PhDS_3\})$ est déclenché à la ligne 10. Dans notre cas, DECA$_{LR}{}^1(I_1, \{Lyc_2 \vee Prim_2, PhDS_3\}) = \{I_1, Univ_2, PhDS_3\}$, et donc on obtient DECA$_{LR}{}^3(PhDS_3, \{Lyc_2 \vee Prim_2\}) = \{PhDS_3, I_1, Univ_2\}$.

En effet, DECA$_{LR}{}^1(I_1, \{Lyc_2 \vee Prim_2, PhDS_3\})$ donne $CI = \{I_1, Univ_2\}$ à la ligne 4 et l'appel récursif DECA$_{LR}{}^2(Univ_2, \{Lyc_2 \vee Prim_2, PhDS_3, I_1\})$ est ainsi déclenché à la ligne 10. Puisque DECA$_{LR}{}^2(Univ_2, \{Lyc_2 \vee Prim_2, PhDS_3, I_1\}) = \{Univ_2, PhDS_3\}$, on obtient DECA$_{LR}{}^1(I_1, \{Lyc_2 \vee Prim_2, PhDS_3\}) = \{I_1, Univ_2, PhDS_3\}$. En effet, DECA$_{LR}{}^2(Univ_2, \{Lyc_2 \vee Prim_2, PhDS_3, I_1\})$ donne $CI = \{Univ_2, PhDS_3\}$ à la ligne 4, l'appel récursif DECA$_{LR}{}^3($ $PhDS_3, \{Lyc_2 \vee Prim_2, PhDS_3, I_1, Univ_2\})$ est ainsi déclenché à la ligne 10. La top clause de cet appel étant dans l'historique, on a que DECA$_{LR}{}^3(PhDS_3, \{Lyc_2 \vee Prim_2, PhDS_3, I_1, Univ_2\}) = \emptyset$ grâce à la condition d'arrêt évitant les dérivations cycliques à la ligne 1. Par conséquent, on obtient DECA$_{LR}{}^2(Univ_2, \{Lyc_2 \vee Prim_2, PhDS_3, I_1\}) = \{Univ_2, PhDS_3\}$.

Nous détaillons maintenant le calcul de DECA$_{LR}{}^3(Dip_3, \{Lyc_2 \vee Prim_2\})$ qui calcule des impliqués du P2PIS avec Dip_3 et $Lyc_2 \vee Prim_2$, y compris tous les impliqués premiers propres à Dip_3.

Il commence par les déductions linéaires locales, ce qui donne $CI = \{Dip_3, L_1 \vee M_1\}$ à la ligne 4. Puisque L_1 et M_1 sont des littéraux dans le langage du pair distant $\mathcal{P}_1$, DECA$_{LR}{}^1(L_1, \{Lyc_2 \vee Prim_2, Dip_3\})$ et DECA$_{LR}{}^1(M_1, \{Lyc_2 \vee Prim_2, Dip_3\})$ sont déclenchés à la ligne 10. Dans notre cas, DECA$_{LR}{}^1(L_1, \{Lyc_2 \vee Prim_2, Dip_3\}) = \{\square\}$ et DECA$_{LR}{}^1(M_1, \{Lyc_2 \vee Prim_2, Dip_3\}) = \{M_1, Col_2\}$ donc on obtient DECA$_{LR}{}^3(Dip_3, \{Lyc_2 \vee Prim_2\}) = \{Dip_3, M_1, Col_2\}$.

En effet, DECA$_{LR}{}^1(L_1, \{Lyc_2 \vee Prim_2, Dip_3\})$ donne $CI = \{L_1, Cher_2, MFac_3\}$ à la ligne 4, l'appel récursif DECA$_{LR}{}^2(Cher_2, \{Lyc_2 \vee Prim_2, Dip_3, L_1\})$ est ainsi déclenché à la ligne 10. DECA$_{LR}{}^2($ $Cher_2, \{Lyc_2 \vee Prim_2, PhDS_3, L_1\})$ donne $CI = \{Cher_2, Fac_3\}$ à la ligne 4,

l'appel récursif $\text{DECA}_{LR}{}^3(Fac_3, \{Lyc_2 \vee Prim_2, Dip_3, L_1, Cher_2\})$ est ainsi déclenché à la ligne 10. $\text{DECA}_{LR}{}^3(\ Fac_3, \{Lyc_2 \vee Prim_2, PhDS_3, L_1, Cher_2\})$ donne $CI = \{\square\}$ à la ligne 4 et donc $\text{DECA}_{LR}{}^2(\ Cher_2, \{Lyc_2 \vee Prim_2, PhDS_3, L_1\}) = \{\square\}$ et on obtient $\text{DECA}_{LR}{}^1(L_1, \{Lyc_2 \vee Prim_2, Dip_3\}) = \{\square\}$.

À son tour, $\text{DECA}_{LR}{}^1(M_1, \{Lyc_2 \vee Prim_2, Dip_3\})$ donne $CI = \{M_1, Col_2\}$ à la ligne 4, l'appel récursif $\text{DECA}_{LR}{}^2(Col_2, \{Lyc_2 \vee Prim_2, Dip_3, M_1\})$ est ainsi déclenché à la ligne 10. $\text{DECA}_{LR}{}^2(\ Col_2, \{Lyc_2 \vee Prim_2, PhDS_3, M_1\})$ donne $CI = \{Col_2\}$ à la ligne 4 et le raisonnement termine du fait qu'il n'y a pas de littéraux de pairs distants. $\text{DECA}_{LR}{}^2(\ Col_2, \{Lyc_2 \vee Prim_2, PhDS_3, M_1\}) = \{Col_2\}$ et donc on obtient $\text{DECA}_{LR}{}^1(M_1, \{Lyc_2 \vee Prim_2, Dip_3\}) = \{Col_2\}$. ■

Propriétés

Le Théorème 1 énonce les propriétés de DECA_{LR}. Ce Théorème établit que DECA_{LR} est une procédure effective pour la déduction linéaire décentralisée dont la top clause est dans le langage du pair interrogé. Il est correct par rapport aux calcul des impliqués, il est complet par rapport au calcul des impliqués premiers propres à la top clause et il termine. Le texte entre crochets est à considérer seulement si on considère le texte entre crochets dans DECA_{LR}.

Le Théorème 1 repose sur la Propriété 3 suivante.

Propriété 3 *Soit $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ un P2PIS et $peer_k$ un pair de $\mathcal{S}$. Étant donné une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, le nombre d'appels récursifs imbriqués et la taille de l'historique sont bornés durant le calcul de $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ par $\#\mathcal{A}(\mathcal{S}) + 1$.*

Preuve. Observons que n'importe quel appel récursif initié par $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ ajoute q_k dans l'historique, alors que tout autre appel récursif initié pendant le calcul de $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ ajoute un seul littéral à l'historique (à cause de la ligne 10).

Montrons maintenant que DECA_{LR} s'arrête avant de pouvoir rajouter à l'historique un littéral qui y est déjà ou son complément.
La condition d'arrêt de la ligne 1 (pour empêcher les raisonnements cycliques) justifie l'arrêt de DECA_{LR} avant de pouvoir rajouter à l'historique un élément qui y est déjà.
Supposons maintenant que le complément $\bar{l}$ d'un littéral l de l'historique s'ajoute dans *hist* à la ligne 10 de DECA_{LR}. Dans ce cas, $\bar{l}$ est utilisé comme top clause pour les dérivations locales produites à la ligne 4. Puisque l appartient à *hist* et *hist* est utilisé dans les dérivations locales, alors $\square$ est dérivé à la ligne 4, ainsi DECA_{LR} s'arrête à la ligne 6. C'est contradictoire avec le fait que DECA_{LR} peut ajouter $\bar{l}$ à *hist* à la ligne 10.
Par conséquent, la taille de l'historique est $\#\mathcal{A}(\mathcal{S})+1$ dans le pire des cas. Dans ce cas, il contient la clause q_k qui a débuté les déductions linéaires décentralisées en plus de A ou $\neg A$ pour chaque variable A de $\mathcal{A}(\mathcal{S})$. Le nombre d'appels récursifs imbriqués est donc limité, par construction de l'historique, à $\#\mathcal{A}(\mathcal{S}) + 1$. □

Theorème 1 *Soit $\mathcal{S}$ un P2PIS dont un pair est $\mathcal{P}_k$. Soit la clause q_k appartenant à $\mathcal{L}(\mathcal{P}_k)$.*

1. $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ *s'arrête,*
2. *si $c \in \text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ alors $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\} \models c$ [et $c \in \mathcal{L}(\mathcal{P}_k)$],et*
3. *si $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\})$, $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \backslash \{q_k\} \not\models c$ [, et $c \in \mathcal{L}(\mathcal{P}_k)$] alors $c \in \text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$.*

Preuve. Le premier point 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 contradictoire avec la Propriété 3 qui établit que le nombre maximal d'appels récursifs imbriqués est fini (par construction de DECA$_{LR}$).

Le second point est prouvé en montrant que tout appel DECA$_{LR}{}^{i}(q_i, hist[,j])$ dérive des impliqués de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$. Ceci est montré par récurrence sur le nombre d'appels ar à DECA$_{LR}$ pour dériver un impliqué c.

Si $ar = 0$, soit $q_i \in hist$ à la ligne 1, soit $\square \in CI$ à la ligne 5, soit il n'y a pas de clause contenant des littéraux de pairs distants à la ligne 9.

- Si $q_i \in hist$, l'algorithme ne retourne pas d'impliqué.
- Si $\square \in CI$, $\square$ est effectivement un impliqué de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$, puisque c'est un impliqué de $\mathcal{T}(\mathcal{P}_i) \cup \mathcal{M}(\mathcal{P}_i) \cup hist \cup \{q_i\} \subseteq \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$.
- S'il n'y a pas de clause contenant des littéraux de pairs distants à la ligne 9, seules les clauses calculées localement à la ligne 4 [ligne 8] sont retournées. Ce sont des impliqués de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$ [$\in \mathcal{L}(\mathcal{P}_j)$] car elles sont produites par déduction linéaire à partir de $\mathcal{T}(\mathcal{P}_i) \cup \mathcal{M}(\mathcal{P}_i) \cup hist \cup \{q_i\} \subseteq \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$.

Supposons maintenant que l'hypothèse de récurrence est vraie pour $ar < n$.

Si $ar = n$, tout appel DECA$_{LR}{}^{i}(q_i, hist[,j])$ déclenché pour produire c déclenche au plus $n-1$ appels à DECA$_{LR}{}^{l}(q_l, hist \cup \{q_i\}[,j])$. Par hypothèse de récurrence, ces appels produisent des impliqués de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i, q_l\}$. Par les Propositions 1 et 2, nous avons alors que DECA$_{LR}{}^{i}(q_i, hist[,j])$ produit des impliqués de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$.

Concernant le troisième point, puisque $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup \{q_k\})$, $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \backslash \{q_k\} \not\models c$ [et $c \in \mathcal{L}(\mathcal{P}_k)$] alors $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ (complétude de la déduction linéaire pour le calcul de conséquences). Nous montrons alors que $c \in$ DECA$_{LR}{}^{k}(q_k, hist[,k])$ par récurrence sur le nombre de décomposions d de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash_{LR}^{q_k} c$ selon les Propositions 1 et 2.

Si $d = 0$ alors toutes les clauses de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \vdash_{LR}^{q_k} c$ sont des clauses de $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k) \cup hist \cup \{q_k\}$ et $c \in$ DECA$_{LR}{}^{k}(q_k, hist[,k])$ par complétude de la déduction linéaire pour le calcul de conséquences (ligne 4).

Supposons que $c \in$ DECA$_{LR}{}^{k}(q_k, hist[,k])$ pour $d < n$.

Si $d = n$ alors tout appel récursif DECA$_{LR}{}^{i}(q_i, hist \cup \{q_k\}[,k])$ déclenche à son tour moins de n appels à DECA$_{LR}$. Par hypothèse de récurrence, un tel appel produit les impliqués premiers propres de q_i par rapport à $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_k\}$. Par conséquent, $c \in$ DECA$_{LR}{}^{k}(q_k, hist[,k])$ du fait de l'hypothèse de récurrence et des Propositions 1 et 2. □

Complexités

Complexité communicationnelle Un aspect important des algorithmes décentralisés est leur besoin de mécanismes de communication, afin que les pairs coopèrent au sein du P2PIS. Il est alors important d'étudier le coût de communication de ces algorithmes dans le pire des cas.

La Proposition 3 fournit une borne supérieure de la complexité de communication en termes du nombre de messages échangés au sein du P2PIS pour calculer DECA$_{LR}{}^{k}(q_k, \emptyset[,k])$.

Proposition 3 *Soient $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..m}$ un P2PIS et $\mathcal{P}_k$ un pair de $\mathcal{S}$. Soit n le nombre maximal de littéraux distincts de pairs distants apparaissant dans l'ensemble des mappings de n'importe*

quel pair. Étant donné une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$, une borne supérieure du nombre de messages échangés au sein de $\mathcal{S}$ afin de calculer $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ *est* $2 * \Sigma_{i=1}^{\#\mathcal{A}(\mathcal{S})+1} n^i$.

Preuve. La Proposition 3 découle des deux observations suivantes.

Tout d'abord, n'importe quel appel (récursif) à DECA_{LR} peut déclencher dans le pire des cas un nombre d'appels récursifs qui est égal au nombre de littéraux *distincts* des pairs distants apparaissant dans les clauses de CI aux lignes 9 et 10. Nous supposons ici l'implémentation d'une optimisation simple (la gestion d'un cache) consistant à faire un *seul* appel récursif par littéral distinct d'un pair distant apparaissant dans les clauses de CI aux lignes 9 et 10, au lieu de faire (naïvement) un appel récursif pour chaque occurrence d'un tel littéral dans les clauses de CI. Ainsi, dans la Proposition 3, n est une borne supérieure du nombre d'appels récursifs pouvant être déclenchés par n'importe quel appel (récursif) à DECA_{LR}.

Ensuite, $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ peut déclencher au pire des cas un nombre d'appels récursifs *imbriqués* qui est borné par $\#\mathcal{A}(\mathcal{S}) + 1$ (*cf.* la Propriété 3).

Par conséquent, la borne supérieure du coût de communication dans le pire des cas de $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ correspond au cas où $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ déclenche n appels récursifs, et dont chaque appel récursif déclenche à son tour n appels récursifs, et ainsi de suite jusqu'à atteindre le nombre maximal d'appels récursifs imbriqués $\#\mathcal{A}(\mathcal{S}) + 1$. Dans ce cas, le nombre d'appels récursifs déclenchés par $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ correspond au nombre d'arêtes dans un arbre n-aire parfait de profondeur $\#\mathcal{A}(\mathcal{S}) + 1$ (voir la Figure 2.4), c-à-d. $\Sigma_{i=1}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ arêtes. Un appel récursif d'un pair à un autre modélise deux messages entre ces deux pairs : un message pour la question et un autre pour les réponses correspondantes. La borne supérieure du coût de communication de $\text{DECA}_{LR}{}^k(q_k, \emptyset[, k])$ dans le pire des cas est donc $2 * \Sigma_{i=1}^{\#\mathcal{A}(\mathcal{S})+1} n^i$.

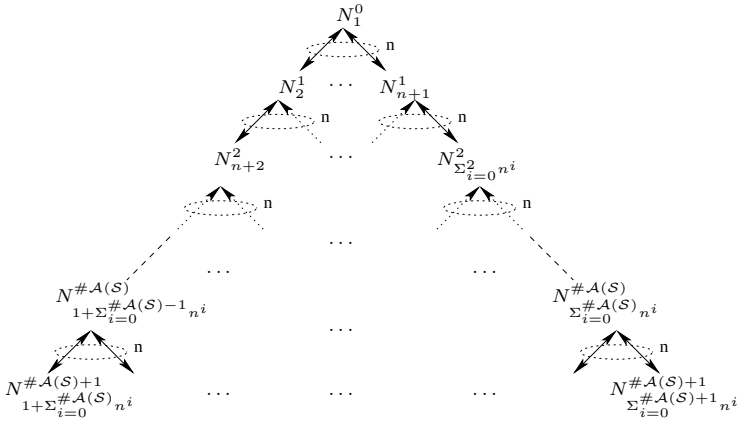

FIGURE 2.4 – Graphe d'un arbre n-aire parfait de profondeur $\#\mathcal{A}(\mathcal{S})+1$. le nom N_p^d d'un noeud signifie que le noeud est à une profondeur d et position (largeur d'abord) p dans l'arbre. Un tel arbre a $\Sigma_{i=0}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ noeuds et $(\Sigma_{i=0}^{\#\mathcal{A}(\mathcal{S})+1} n^i) - 1$ arêtes, c-à-d. $\Sigma_{i=1}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ arêtes. □

Complexité de DECA_{LR} **en espace et en temps** Dans un algorithme décentralisé, la complexité en espace reflète l'espace maximal utilisé simultanément *par le P2PIS*, c-à-d. distribué sur plusieurs pairs, afin d'obtenir les résultats d'un certain calcul. Quant à la complexité en temps, elle reflète le temps maximal afin d'obtenir les résultats d'un certain calcul.

La Propriété suivante est utile pour établir et comprendre les complexités de DECA$_{LR}$ en espace et en temps. Elle fournit à la fois une borne supérieure pour le nombre de clauses dans une théorie et le nombre des impliqués premiers de cette théorie.

Propriété 4 *Soit $\mathcal{T}$ une théorie clausale de LP sans tautologies et sans redondance de littéraux en terme d'un alphabet propositionnel $\mathcal{A}$. On a :*
- $\#\mathcal{T} \leq 3^{\#\mathcal{A}}$ *et*
- $\#\mathbf{PI}(\mathcal{T}) < 2^{\#\mathcal{T}} < 3^{\#\mathcal{A}}$.

Cette Propriété découle de la Propriété 1 et du fait qu'on peut générer toutes les clauses en terme de $\mathcal{A}$ qui ne sont pas des tautologies et sans redondance de littéraux, donc un sur-ensemble des impliqués premiers de $\mathcal{T}$, en calculant $\otimes_{V\in\mathcal{A}}\{\Box, V, \neg V\}$.

Remarquons que, si on regroupe les deux points de la Propriété 4, $3^{\#\mathcal{A}}$ est en général exponentiellement plus grande que $\#\mathcal{T}$.

Passons maintenant au nombre de clauses à stocker simultanément (pour l'espace) et le nombre de clauses à générer (pour le temps) afin de calculer DECA$_{LR}{}^k(q_k, \emptyset[, k])$. Dans la suite, nous faisons deux suppositions sur l'implémentation de DECA$_{LR}$. Premièrement, nous adoptons encore l'optimisation de mise en cache, comme dans la complexité communicationnelle. Deuxièmement, DECA$_{LR}$ peut utiliser deux stratégies pour déclencher les appels récursifs : la stratégie *en profondeur d'abord* ou la stratégie *en parallèle*. Comme leurs noms indiquent, la première stratégie déclenche les appels récursifs l'un à la suite de l'autre, tandis que la deuxième déclenche les appels en parallèle. Évidemment, la stratégie en profondeur d'abord économise de l'espace et prend plus de temps, tandis que la stratégie en parallèle prend plus d'espace et moins de temps.

La Proposition 4 établit que DECA$_{LR}{}^k(q_k, \emptyset)$ peut exiger, avec les deux stratégies, un espace et un temps exponentiels dans la taille du P2PIS. Ce résultat n'est pas surprenant du fait que DECA$_{LR}$ calcule tous les impliqués premiers qui découlent de la top clause, donc tous les impliqués premiers du P2PIS dans le pire des cas et le nombre de ces impliqués premiers peut être exponentiel dans la taille du P2PIS plus la top clause (Propriété 1). Quant à DECA$_{LR}{}^k(q_k, \emptyset, k)$, il peut exiger un espace exponentiel dans la taille d'*un pair* et un temps exponentiel dans la taille du P2PIS avec la stratégie en profondeur d'abord, et un espace exponentiel dans la taille du P2PIS et un temps exponentiel dans la taille d'*un pair* avec la stratégie en en parallèle.

Proposition 4 *Soient un P2PIS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..m}$ et un pair $\mathcal{P}_k$ de $\mathcal{S}$. Soit a la taille maximale des alphabets des pairs : $a = max(\mathcal{A}(\mathcal{P}_1), \ldots, \mathcal{A}(\mathcal{P}_m))$. Soit n le nombre maximal de littéraux distincts de pairs distants apparaissant dans l'ensemble des mappings de n'importe quel pair. Étant donné une clause $q_k \in \mathcal{L}(\mathcal{P}_k)$ avec q littéraux.*

(i) Une borne supérieure de l'espace nécessaire pour calculer
- DECA$_{LR}{}^k(q_k, \emptyset)$ *est $O(\#\mathcal{A}(\mathcal{S})^2 * n * 3^{\#\mathcal{A}(\mathcal{S})})$ clauses avec la stratégie en profondeur d'abord et $O(n^{\#\mathcal{A}(\mathcal{S})+3} * 3^{\#\mathcal{A}(\mathcal{S})})$ clauses avec la stratégie en parallèle.*
- DECA$_{LR}{}^k(q_k, \emptyset, k)$ *est $O(\#\mathcal{A}(\mathcal{S})^2 * n * 3^{a+n+q})$ clauses avec la stratégie en profondeur d'abord et $O(n^{\#\mathcal{A}(\mathcal{S})+3} * 3^{a+n+q})$ clauses avec la stratégie en parallèle.*

(ii) Une borne supérieure du temps nécessaire pour calculer
- DECA$_{LR}{}^k(q_k, \emptyset)$ *est $O(n^{\#\mathcal{A}(\mathcal{S})+2} * 3^{\#\mathcal{A}(\mathcal{S})*n})$ clauses à générer avec la stratégie en profondeur d'abord et $O(\#\mathcal{A}(\mathcal{S}) * 3^{\#\mathcal{A}(\mathcal{S})*n})$ clauses à générer avec la stratégie en parallèle.*
- DECA$_{LR}{}^k(q_k, \emptyset, k)$ *est $O(n^{\#\mathcal{A}(\mathcal{S})+2} * 3^{a*n+q})$ clauses à générer avec la stratégie en profondeur d'abord et $O(\#\mathcal{A}(\mathcal{S}) * 3^{a*n+q})$ clauses à générer avec la stratégie en parallèle.*

Preuve. Concernant la complexité en espace, considérons tout d'abord l'espace *local* nécessaire pour produire un appel (récursif) $\text{DECA}_{LR}{}^i(q_i, hist[,k])$ (ici i peut être k). DECA_{LR} a besoin de stocker q_i, les clauses de $hist$ (bornées par $\#\mathcal{A}(\mathcal{S})+1$, Propriété 3), et les clauses de $\mathcal{T}(\mathcal{P}_i)\cup\mathcal{M}(\mathcal{P}_i)$ (bornées par toutes les clauses non tautologiques sans redondance de littéraux pouvant être générées en utilisant les variables de $\mathcal{A}(\mathcal{P}_i)$ et les variables des pairs distants apparaissant dans $\mathcal{M}(\mathcal{P}_i)$). En tout, on a au plus $\#\mathcal{A}(\mathcal{S}) + 2 + 3^{a+n}$ clauses à stocker localement.
Passons maintenant à l'unique variable locale CI de DECA_{LR}. À la ligne 4, CI contient des impliqués de $\mathcal{T}(\mathcal{P}_i) \cup \mathcal{M}(\mathcal{P}_i) \cup hist \cup \{q_i\}$, y compris tous les premiers propres à q_i. La taille de CI est alors bornée par le nombre des clauses non tautologiques sans redondance de littéraux pouvant être générées en utilisant les variables de $\mathcal{A}(\mathcal{P}_i)$ (bornées par a dans la Proposition 4), les variables des pairs distants apparaissant dans $\mathcal{M}(\mathcal{P}_i)$ (bornées par n dans la Proposition 4) et les variables de la clause $q_k \in hist$ (q dans la Proposition 4) qui a initié la déduction linéaire. Les autres clauses de $hist$ étant des littéraux (par construction de l'historique), elles ne peuvent pas fournir de nouvelles variables dans une déduction linéaire de q_i, donc dans CI. Il en découle que CI a une taille maximale de 3^{a+n+q} à la ligne 4.
Après la ligne 10, CI contient des impliqués de $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \cup hist \cup \{q_i\}$, y compris tous les impliqués premiers propres à q_i. Quand l'appel à DECA_{LR} est $\text{DECA}_{LR}{}^i(q_i, hist)$, resp. $\text{DECA}_{LR}{}^i(q_i, hist, k)$, la taille de CI est bornée par le nombre des clauses non tautologiques sans redondance de littéraux pouvant être générées en utilisant les variables du P2PIS, resp. de $\mathcal{A}(\mathcal{P}_k)$. Par conséquent, après la ligne 10, la taille de CI est au plus $3^{\#\mathcal{A}(\mathcal{S})}$ clauses (contenant les 3^{a+n+q} clauses stockées à la ligne 4), resp. $3^{a+n+q} + 3^a$ clauses. De plus, le calcul effectué aux lignes 9 et 10 nécessite le stockage des résultats des appels récursifs à DECA_{LR} (bornés par n à cause de notre supposition de "caching"). Puisque ces appels fournissent des impliqués du P2PIS, chacun exige au plus un stockage de $3^{\#\mathcal{A}(\mathcal{S})}$ clauses, resp. 3^a clauses. On obtient donc, pour le moment, l'espace nécessaire pour un appel (récursif) à DECA_{LR} est au plus $[\alpha]$ $\#\mathcal{A}(\mathcal{S}) + 2 + 3^{a+n} + (n+1) * 3^{\#\mathcal{A}(\mathcal{S})}$ clauses pour $\text{DECA}_{LR}{}^i(q_i, hist)$ ou $[\beta]$ $\#\mathcal{A}(\mathcal{S}) + 2 + 3^{a+n} + 3^{a+n+q} + (n+1) * 3^a$ pour $\text{DECA}_{LR}{}^i(q_i, hist, k)$.

Passons maintenant au calcul de $\text{DECA}_{LR}{}^k(q_k, hist)$ et $\text{DECA}_{LR}{}^k(q_k, hist, k)$ dans le P2PIS. Dans le pire des cas, $\Sigma_{i=0}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ appels (récursifs) sont déclenchés (voir le pire des cas de la complexité communicationnelle de DECA_{LR}). Selon la stratégie adoptée par DECA_{LR} pour déclencher les appels récursifs, on a un nombre d'appels actifs maximal de $[\gamma]$ $(\#\mathcal{A}(\mathcal{S}) + 1)$ avec la stratégie en profondeur d'abord et au plus $[\delta]$ $\Sigma_{i=0}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ ($< n^{\#\mathcal{A}(\mathcal{S})+2}$ en supposant que $n \geq 2$ dans un P2PIS) avec la stratégie en parallèle.

On obtient ainsi les bornes suivantes pour la complexité en espace de DECA_{LR} :

- de $[\alpha]$ et $[\gamma]$, on a besoin d'un stockage simultané dans $O(\#\mathcal{A}(\mathcal{S})^2 * n * 3^{\#\mathcal{A}(\mathcal{S})})$ clauses afin de calculer $\text{DECA}_{LR}{}^k(q_k, hist[,k])$ avec la stratégie en profondeur d'abord.
- De $[\alpha]$ et $[\delta]$, on a besoin d'un stockage simultané qui est au plus dans $O(n^{\#\mathcal{A}(\mathcal{S})+3} * 3^{\#\mathcal{A}(\mathcal{S})})$ clauses afin de calculer $\text{DECA}_{LR}{}^k(q_k, hist[,k])$ avec la stratégie en parallèle.
- De $[\beta]$ et $[\gamma]$, on a besoin d'un stockage simultané qui est au plus dans $O(\#\mathcal{A}(\mathcal{S})^2 * n * 3^{a+n+q})$ clauses afin de calculer $\text{DECA}_{LR}{}^k(q_k, hist, k)$ avec la stratégie en profondeur d'abord.
- De $[\beta]$ et $[\delta]$, on a besoin d'un stockage simultané qui est au plus dans $O(n^{\#\mathcal{A}(\mathcal{S})+3} * 3^{a+n+q})$ clauses afin de calculer $\text{DECA}_{LR}{}^k(q_k, hist, k)$ avec la stratégie en parallèle.

Pour la complexité en temps de DECA_{LR}, nous suivons le même raisonnement.

Considérons de le temps *local* nécessaire pour effectuer n'importe quel appel (récursif)

DECA$_{LR}{}^i(q_i, hist[,k])$ (i peut être k) déclenché durant le calcul de DECA$_{LR}{}^k(q_k, \emptyset[,k])$. On se concentre ici sur la borne supérieure du nombre de clauses à générer et *non* sur le temps nécessaire pour les générer, car dans le second cas il aurait fallu choisir une implémentation précise de $\vdash_{LR}$ et de $\circledcirc$.

À la ligne 4, au plus 3^{a+n+q} clauses sont générées en utilisant $\vdash_{LR}$ (voir la complexité en espace de DECA$_{LR}$). De plus, aux lignes 9 et 10, au plus $3^{\#\mathcal{A}(\mathcal{S})*n}$ clauses sont générées avec $\circledcirc$ pour DECA$_{LR}{}^i(q_i, hist)$ et au plus 3^{a*n} clauses sont générées avec $\circledcirc$ pour DECA$_{LR}{}^i(q_i, hist, k)$. En effet, le nombre de littéraux de pairs distants dans n'importe quel mapping est borné par n et les tailles des sorties des appels récursifs sont bornées par $3^{\#\mathcal{A}(\mathcal{S})}$ pour DECA$_{LR}{}^i(q_i, hist)$ et par 3^a pour DECA$_{LR}{}^i(q_i,\ hist, k)$ (voir la complexité en espace de DECA$_{LR}$). Pour le moment, on obtient que le temps local d'un appel (récursif) à DECA$_{LR}$ est au plus $[\alpha]\ 3^{a+n+q} + 3^{\#\mathcal{A}(\mathcal{S})*n}$ clauses générées pour DECA$_{LR}{}^i(q_i, hist)$ ou $[\beta]\ 3^{a+n+q} + 3^{a*n}$ clauses générées pour DECA$_{LR}{}^i(q_i, hist, k)$.

Passons maintenant au calcul de DECA$_{LR}{}^k(q_k, hist)$ et DECA$_{LR}{}^k(q_k, hist, k)$ au sein du P2PIS. Dans le pire des cas, $\Sigma_{i=0}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ appels (récursifs) sont déclenchés (voir le pire des cas de la complexité communicationnelle de DECA$_{LR}$). Si on considère la stratégie en profondeur d'abord pour déclencher les appels récursifs, une borne supérieure du temps total nécessaire correspond à faire la somme du temps nécessaire pour $[\gamma]\ \Sigma_{i=0}^{\#\mathcal{A}(\mathcal{S})+1} n^i$ appels (récursifs) ($< n^{\#\mathcal{A}(\mathcal{S})+2}$ avec la supposition que $n \geq 2$ dans un P2PIS). Au contraire, et si on considère la stratégie en parallèle, une borne supérieure du temps total nécessaire correspond à faire la somme des temps nécessaires pour le nombre maximal des appels récursifs *imbriqués* $[\delta]\ \#\mathcal{A}(\mathcal{S}) + 1$ (Propriété 3).

On obtient alors les bornes supérieures suivantes de la complexité en temps de DECA$_{LR}$:

- de $[\alpha]$ et $[\gamma]$, on a besoin de générer au plus un nombre de clauses $O(n^{\#\mathcal{A}(\mathcal{S})+2} * 3^{\#\mathcal{A}(\mathcal{S})*n})$ afin de calculer DECA$_{LR}{}^k(q_k, hist[,k])$ avec la stratégie en profondeur d'abord.
- de $[\alpha]$ et $[\delta]$, on a besoin de générer au plus un nombre de clauses $O(\#\mathcal{A}(\mathcal{S}) * 3^{\#\mathcal{A}(\mathcal{S})*n})$ afin de calculer DECA$_{LR}{}^k(q_k, hist[,k])$ avec la stratégie en parallèle.
- de $[\beta]$ et $[\gamma]$, on a besoin de générer au plus un nombre de clauses qui est $O(n^{\#\mathcal{A}(\mathcal{S})+2} * 3^{a*n+q})$ afin de calculer DECA$_{LR}{}^k(q_k, hist, k)$ avec la stratégie en profondeur d'abord.
- de $[\beta]$ et $[\delta]$, on a besoin de générer au plus un nombre de clauses qui est $O(\#\mathcal{A}(\mathcal{S}) * 3^{a*n+q})$ afin de calculer DECA$_{LR}{}^k(q_k, hist, k)$ avec la stratégie en parallèle.

□

2.3 Travaux connexes

Nous avons proposé dans le chapitre un premier algorithme décentralisé de calcul de conséquences – en logique propositionnelle – fondé sur la déduction *linéaire*. Cependant, des algorithmes décentralisés de calcul de conséquences fondé sur la déduction (générale) ont été déjà proposés dans la littérature.

2.3.1 Calcul de conséquences et optimisation

La décentralisation du calcul de conséquences en logique propositionnelle a tout d'abord été étudié sous l'angle de l'optimisation.

Étant donné une théorie (clausale) centralisée de LP dans laquelle il faut effectuer un calcul de conséquences, [10, 11, 12] proposent de commencer par partitionner la théorie en plusieurs

théories suivant une *structure d'arbre*. Ensuite, un algorithme décentralisé de déduction appelé Message-Passing algorithm (MP) exploite astucieusement son partitionnement arborescent afin d'effectuer un calcul efficace.

Plus précisément, la théorie partitionnée induit un graphe appelé *graphe d'intersection* où chaque noeud représente une sous-théorie du partitionnement ; deux noeuds sont liés par une arête s'ils ont des variables propositionnelles en commun ; une arête entre deux noeuds est étiquetée avec toutes leurs variables communes. Le calcul de conséquences est effectué en parallèle dans chaque sous-théorie en utilisant la stratégie de résolution. Le transfert de formules entre les sous-théories est contrôlé par les étiquettes des arêtes du graphe d'intersection : les conséquences logiques trouvées au sein d'une sous-théorie sont transférées à une autre sous-théorie (et sont ajoutés à l'ensemble de ses formules) *seulement si* elles utilisent uniquement des variables propositionnelles qui sont partagées entre ces deux sous-théories.

Pour être complet pour le calcul de conséquences, cet algorithme doit être appliqué à un graphe d'intersection sans cycle. L'idée est que tout graphe peut être transformé en temps polynomial en un graphe sans cycle avec des étiquettes élargies et garantir par la suite la complétude. L'algorithme BREAK-CYCLES décrit dans [12] réalise la transformation appropriée.

Discussion 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. On peut alors envisager de l'utiliser pour la déduction dans des P2PIS propositionnel. Toutefois, la limitation sévère à son utilisation dans des applications réelles est que les P2PIS *doivent* avoir une topologie d'arbre.

2.3.2 Calcul de conséquences dans un P2PIS propositionnel

Le premier calcul de conséquences totalement décentralisé en LP a été proposé dans [8]. Ce calcul s'effectue dans un P2PIS propositionnel composé de pairs autonomes gérant chacun une théorie en LP en terme de son propre langage. Les pairs peuvent être sémantiquement connectés en ayant des variables en commun, appelées variables partagées. Aucun pair n'a la connaissance de la théorie globale du P2PIS. Chaque pair connaît seulement sa théorie locale et sait qu'il partage certaines variables de son vocabulaire avec d'autres pairs (son *voisinage*). Le P2PIS peut être donc représenté par un *graphe de voisinage* avec des noeuds et des arêtes étiquetées, où les noeuds représentent les théories des pairs et une arête étiquetée entre deux pairs exprime que les pairs savent tous deux qu'ils partagent les variables portées par l'étiquette.

Pour calculer les conséquences d'un P2PIS et une clause fournie en entrée à un pair, [8] propose l'algorithme DeCA (DEcentralized Consequence Algorithm) qui est le premier algorithme spécialement conçu pour faire de la déduction décentralisée dans des théories propositionnelles distribuées. Notamment, DeCA n'impose aucune contrainte sur la topologie du P2PIS.

DeCA calcule – en suivant la stratégie de résolution générale (c-à-d. sans restriction de l'application de la règle de résolution) – des impliqués du P2PIS et d'une clause fournie à un pair, y compris tous les impliqués premiers propres de la clause. Il est prouvé correct et complet pour le calcul de conséquences.

Des expérimentations ont été faites dans [7] pour le passage à l'échelle de DeCA. Les expérimentations sont essentiellement sur des P2PIS formés de clauses KROM (c-à-d. des clauses ayant deux littéraux) et de 3-clauses (c-à-d. clauses avec au plus 3 littéraux) allant jusqu'à mille pairs. DeCA montre un très bon comportement dans les P2PIS avec des KROM clauses, alors qu'il a besoin parfois d'un mécanisme de *time out* pour terminer dans les P2PIS avec des 3-clause.

Discussion Tout comme DeCA, DeCA$_{LR}$ n'impose aucune restriction sur la topologie d'un P2PIS. En fait, DeCA$_{LR}$ est à DeCA ce que la déduction linéaire est à la déduction. Il offre le même pouvoir de déduction, mais il limite le nombre de déductions lorsque l'on veut calculer les impliqués (premiers) *propres* à une clause donnée.

Ainsi, en se basant sur les expérimentations de DeCA, notre algorithme devrait se comporter au moins aussi bien que DeCA dans les P2PIS avec des clauses KROM et les P2PIS avec 3-clauses allant jusqu'à un mille pairs. Ce résultat doit – évidemment – être confirmé par une étude expérimentale dans l'esprit de celle de [7].

Notons que cette thèse fournie la première étude de complexité (communicationnelle, en espace et en temps) pour un algorithme totalement décentralisé de déduction. Les algorithme de la littérature (par exemple MP et DeCA) n'ont été jusqu'à présents évalués qu'expérimentalement.

2.3.3 Calcul de conséquences dans un P2PIS propositionnel inconsistant

Calculer des conséquences dans de tels systèmes n'a que peu d'intérêt, puisqu'ils n'admettent qu'un unique impliqué premier $\square$. [31] et [21, 22] étendent les travaux sur DeCA en considérant des P2PIS inconsistants : ils se focalisent sur le calcul de conséquences à partir de sous-systèmes consistants.

[31] propose les algorithmes P2P-NG et WF-DeCA pour faire du calcul de conséquences en présence d'inconsistance. P2P-NG et WF-DeCA supposent que les théories des pairs sont consistantes mais que la théorie globale peut être inconsistante. Vu que les théories des pairs sont supposées consistantes, une inconsistance dans la théorie globale est certainement causée par les mappings. Une notion de *nogood* est alors définie comme étant un ensemble de mappings tel que $O \cup ng$ est inconsistant, où ng est un nogood et O est l'union des théories des pairs.

L'algorithme P2P-NG sert à détecter les nogoods. C'est une adaptation de DeCA qui cherche **toutes** les preuves de la clause vide impliqué d'une clause. Avant d'ajouter un nouveau mapping dans le réseau, le pair utilise P2P-NG pour vérifier si ce mapping peut être la cause d'une inconsistance, c-à-d. si la clause vide est un impliqué de ce mapping. Dans ce cas, le pair stocke localement l'ensemble formé par ce nouveau mapping et les autres mapping utilisés dans la preuve de la clause vide (son *mapping support*) comme un nogood.
L'algorithme P2P-NG termine, est correct et complet, dans le sens où tous les nogoods causés par l'ajout d'un nouveau mapping sont détectés. À partir de la définition de nogoods, on sait que tout sous-ensemble du réseau global qui ne contient aucun nogood est forcément consistant. Donc, un impliqué n'est renvoyé que s'il peut être déduit avec au moins une preuve qui ne contient pas de nogood. Un tel impliqué est considéré *bien-fondé* car il est déduit à partir d'un sous-ensemble consistant du réseau global.

Pour ne chercher que des impliqués bien-fondés découlant d'une clause, l'algorithme WF-DeCA collecte *tous* les nogoods qui pourraient invalider une preuve d'un impliqué. Comme un nogood est un ensemble de mappings, WF-DeCA effectue des tests d'inclusion entre le mapping support d'une preuve et les nogoods collectés. Si le mapping support contient un des nogoods collectés, la preuve n'est pas bien-fondée et donc rejetée. WF-DeCA est aussi une adaptation de DeCA, il est utilisé au moment du calcul des impliqués d'une clause, il termine et il est correct pour le calcul d'impliqués bien fondés. Sa correction dépend de la complétude de P2P-NG.

Exemple. 7 (Raisonnement en présence d'inconsistance)
Considérons le P2PIS illustré dans la Figure 2.5. Le P2PIS est inconsistant à cause de l'ensemble

des clauses $\{\neg a_1 \vee \neg b_1, \neg a_2 \vee a_1, a_2, \neg b_3 \vee b_1, b_3\}$. Parmi elles se trouvent deux mappings $P_2.1$ et $P_3.1$. Si on suppose que les mappings se sont ajoutés dans les pairs dans cet ordre : $P_2.1$, $P_3.1$, $P_4.1$. Il est possible de vérifier que l'ajout du mapping $P_2.1$ ne cause pas d'inconsistance. Par contre, quand le mapping $P_3.1$ se rajoute dans P_3, un nogood $ng = \{P_2.1, P_3.1\}$ est détecté et stocké dans P_3. L'addition de $P_4.1$ ne cause pas de nouveau nogood.
Considérons maintenant la demande de calcul des conséquences de q_4 posé à P_4. L'ensemble R des réponses bien fondées retournées par WF-DeCA est $R = \{q_4, a_4 \vee a_1 \vee b_1, a_4 \vee \neg b_1 \vee b_1, a_4 \vee \neg b_3 \vee b_1, a_4 \vee b_1, a_4 \vee a_1 \vee \neg a_1, a_4 \vee a_1 \vee \neg a_1, a_4 \vee a_1, a_4 \vee a_1 \vee b_1 \vee \neg a_1, a_4 \vee a_1 \vee \neg a_1 \vee b_1 \vee \neg b_1\}$. Pour la même demande, DeCA retourne en plus de R un ensemble de clauses tel $a_4, a_4 \vee a_1 \vee b_1 \vee \neg a_2, \ldots$ En fait, ces clauses ne sont pas retournées par WF-DeCA car leurs mappings support contiennent le nogood ng et donc ce ne sont pas des réponses bien fondées.

■

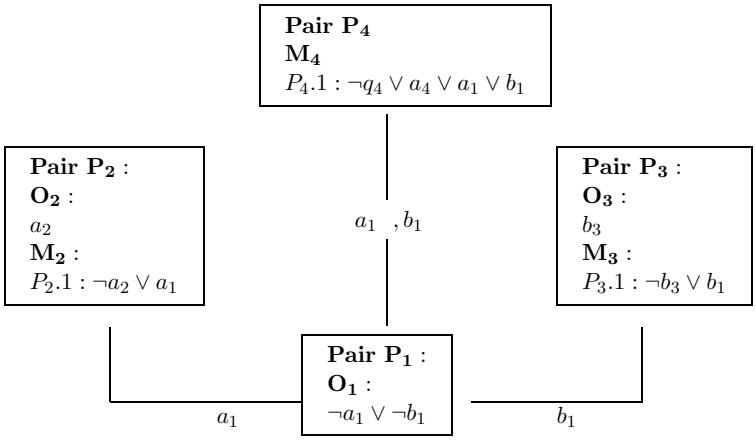

FIGURE 2.5 – Un P2PIS inconsistant.

[21, 22] traite le calcul de conséquences dans une théorie décentralisée propositionnelle inconsistante (afin de répondre à des requêtes par réfutation). En effet, ils observent que dans [31], une formule et sa négation peuvent être à la fois des conséquences bien fondées d'une théorie décentralisée. Ils décident alors de se diriger vers les techniques d'argumentation afin de calculer les conséquences ayant plus de raisons d'être vérifiées que leurs négations et ceci en attribuant des priorités aux pairs selon leur pertinence pour répondre aux requêtes.

Discussion Ces travaux visent à raisonner en présence d'inconsistance. Une alternative serait de "réparer" l'inconsistance. Dans cette thèse, comme nous allons voir dans la Section 3.2 du chapitre 3, nous proposerons un algorithme de test d'extension conservative d'un pair dans un P2PIS. Une utilisation particulière de cet algorithme est la détection des causes de l'inconsistance. Ceci permettra d'envisager des stratégies auto-guérissantes pour restaurer la consistance globale.

Chapitre 3

Test et calcul d'extension non conservative

Nous présentons ici la seconde contribution de cette thèse : un algorithme décentralisé de test et de calcul d'extension non-conservative d'un pair dans un P2PIS propositionnel.

Formellement, étant données deux théories 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).

Dans un P2PIS (indépendamment de la logique utilisée), un pair *étend* sa théorie (à l'aide de mappings) avec celles des autres pairs (et leurs mappings) de sorte à pouvoir utiliser leurs connaissances pour répondre aux questions qui lui sont posées. La notion d'extension conservative est importante ici puisqu'il est notoire que l'extension d'une théorie logique n'est pas conservative dans le cas général. Ainsi, lorsqu'un pair étend sa théorie 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 suivants.

1. Supposons qu'un pair ait une description complète de son domaine d'application au sein de sa théorie, 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 théorie 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 théorie. 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 théorie

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 théorie et aucune expertise particulière à propos de ce domaine. Dans ce cas, la non conservativité de sa théorie 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 1. et 2. ci-dessus sont étroitement liés aux notions de *confidentialité* et de *qualité de service*. 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.

Nous étudions dans ce Chapitre les problèmes consistant, d'une part, à *décider si un P2PIS propositionnel est une extension conservative d'un pair donné* (problème de décision) et, d'autre part, à *calculer les conséquences d'un P2PIS propositionnel exhibant la non conservativité d'un pair donné* (problème fonctionnel).

Le Chapitre est organisé comme suit : nous abordons la notion d'extension (non) conservative dans les P2PIS propositionnels dans la Section 3.1. Nous discutons brièvement de la notion d'extension conservative dans les P2PIS insatisfiables dans la Section 3.2. Nous finissons par un état de l'art sur le domaine dans la Section 3.3.

3.1 Extension (non) conservative d'un pair dans un P2PIS propositionnel

Nous considérons dans ce Chapitre les P2PIS propositionnels (clausaux) du Chapitre précédent.

La notion d'*extension conservative d'un pair dans un P2PIS* est fondée sur la notion de *conséquence* logique.

Définition 4 (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 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 Craig [34], pourront remarquer le lien étroit existant avec la notion d'extension conservative (par exemple [35]). 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 $Tgt(\mathcal{L}(\mathcal{P}))$.

3.1.1 Problème de décision

Partant de la définition 4, notre problème de décision $\mathbf{CE}^{dec}$ est le suivant.

Définition 5 (Problème de décision : $\mathbf{CE}^{dec}$)
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 4 – que le problème de décision dans nos P2PIS clausaux est difficile. Il consiste en effet à résoudre (i) des problèmes Π_1^P-complets pour (ii) un nombre exponentiel de clauses : (i) 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 et (ii) il existe dans le pire des cas $3^{\#(\mathcal{A}(\mathcal{P}))}$ clauses dans $\mathcal{L}(\mathcal{P})$, donc de $Tgt(\mathcal{L}(\mathcal{P}))$ qui peuvent être générés en combinant de façon disjonctive soit V, $\neg V$ ou $\square$ pour chaque variable V de $\mathcal{A}(\mathcal{P})$.

Cette intuition est confirmée par le théorème suivant qui établit la complexité exacte de $\mathbf{CE}^{dec}$ dans nos P2PIS. Il précise aussi la borne supérieure de la complexité de $\mathbf{CE}^{dec}$ dans le cas où nous permettons seulement certaines familles de clauses bien connues dans la littérature, tel que les $\mathbf{K}_{\geq \mathbf{3}}$-clauses, c-à-d. les clauses avec au plus une constante donnée $\mathbf{K} \geq 3$ de littéraux, les clauses de Horn (inverses), c-à-d. les clauses avec au plus un littéral positif (ou négatif), et les clauses de KROM, c-à-d. les clauses ayant au plus 2 littéraux.

Theorème 2 (Complexité de $\mathbf{CE}^{dec}$)
Décider $\mathbf{CE}^{dec}$ est :
- *Π_2^P-complet pour les P2PIS clausaux, c-à-d. sans imposer de restriction sur les clauses,*
- *dans Δ_2^P si on considère des P2PIS $\mathbf{K}_{\geq \mathbf{3}}$, c-à-d. constitués de clauses $\mathbf{K}_{\geq \mathbf{3}}$ seulement,*
- *dans Π_1^P si on considère des P2PIS Horn (inverses), c-à-d. constitués de clauses de Horn (inverses) seulement, et*
- *dans P si on considère des P2PIS KROM, c-à-d. constitué de clause de KROM seulement.*

Preuve. Dans les P2PIS clausaux, l'appartenance à la classe Π_2^P découle du fait que le problème complémentaire est dans Σ_2^P : il suffit de deviner une clause sans redondance de littéral en termes de l'alphabet cible de $\mathcal{P}$ et tester en appelant un Σ_1^P- oracle si la Propriété Ψ de la définition 4 n'est pas vérifiée (c-à-d. $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ et $\mathcal{T}(\mathcal{P}) \not\models c$).
L'aspect Π_2^P-difficile résulte d'une réduction polynomiale du problème canonique de Π_2^P qui est la validité d'une formule $QBF_{2,\forall}$ à notre problème d'extension conservative. Une formule $QBF_{2,\forall}$ a la forme $\forall A \exists B \Phi$ où A et B forment une partition des variables de la formule Φ de LP. L'idée ici est que le problème de validité d'une telle formule est aussi Π_2^P-complet quand Φ est une conjonction de clauses ou d'une manière équivalente une théorie clausale (par exemple [33]). Puisque $\forall A \exists B \Phi$ est valide ssi il n'existe pas de conséquences non tautologiques de Φ en termes des variables de A [55], une preuve de Π_2^P-difficulté pour $\mathbf{CE}^{dec}$ revient à construire un P2PIS tel que $\forall A \exists B \Phi$ est valide ssi le P2PIS est une extension conservative d'un pair donné quand il n'y a pas de conséquences non tautologiques de Φ en termes de A. Par exemple, considérons le P2PIS $\mathcal{S}$ de la Figure 3.1 formé de deux pairs $\mathcal{P}_1$ et $\mathcal{P}_2$: $\mathcal{T}(\mathcal{P}_1) = \{\}$ (c-à-d. la conjonction vide, donc $vrai$), $\mathcal{T}(\mathcal{P}_2) = \Phi$, $\mathcal{M}(\mathcal{P}_1) = \mathcal{M}(\mathcal{P}_2) = \bigcup_{i=1}^{n} \{\neg A_1^i \vee A_2^i, \neg A_2^i \vee A_1^i\}$ tel que : $\mathcal{A}(\mathcal{P}_1) = Tgt(\mathcal{A}(\mathcal{P}_1)) = \{A_1^1, \ldots, A_1^n\}$, $\mathcal{A}(\mathcal{P}_2) = Tgt(\mathcal{A}(\mathcal{P}_2)) = A \cup B$, et $A = \{A_2^1, \ldots, A_2^n\}$.

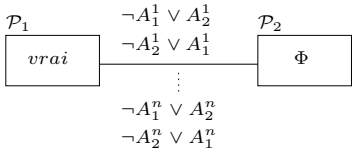

FIGURE 3.1 – Le P2PIS $\mathcal{S}$.

Il découle facilement que $\forall A \exists B \Phi$ est valide (c-à-d. il n'y a pas de conséquence non tautologiques de Φ en termes de A) ssi ce P2PIS est une extension conservative de $\mathcal{P}_1$.

Quand le nombre de littéraux permis dans les clauses d'un P2PIS est borné par une constante $\mathbf{K} \geq 3$, on est dans le fragment $\mathbf{K}_{\geq \mathbf{3}}$-clauses de LP. Dans ce cas, le langage (cible) d'un pair donné contient au plus un nombre de clauses qui est un polynôme (de degré $\mathbf{K}$) dans la taille de son alphabet. Il est bien connu que raisonner avec des $\mathbf{K}_{\geq \mathbf{3}}$-clauses est aussi compliqué que raisonner dans toute la LP. En particulier, décider si une clause est une conséquence d'une théorie $\mathbf{K}_{\geq \mathbf{3}}$-clauses est Π_1^P-complet (rappelons que 3SAT est déjà Σ_1^P-complet [40]). Par conséquent, décider $\mathbf{CE}^{dec}$ revient à générer, dans le pire des cas, un nombre de $\mathbf{K}_{\geq \mathbf{3}}$-clauses qui est polynomial dans la taille de l'alphabet d'un pair donné et pour lequel on teste en appelant un Σ_1^P-oracle la propriété Ψ de la Définition 4. Ainsi, pour les P2PIS $\mathbf{K}_{\geq \mathbf{3}}$, $\mathbf{CE}^{dec}$ est dans $P^{\Sigma_1^P}$ c-à-d. dans Δ_2^P.

Quand le nombre de littéraux permis dans les clauses d'un P2PIS est borné par la constante 2, on se retrouve dans ce qu'on appelle le fragment KROM (ou fragment bijonctif) de LP. Dans ce cas, le langage (cible) d'un pair donné contient au plus un nombre de clauses KROM qui est quadratique dans la taille de son alphabet. De plus, il a été montré que décider si une clause est une conséquence d'une théorie de clauses KROM est dans P (par exemple [38]). Ainsi, décider $\mathbf{CE}^{dec}$ revient à générer, dans le pire des cas, un nombre de clauses KROM qui est polynomial dans la taille de l'alphabet d'un pair donné, puis à tester la Propriété Ψ de la définition 4 pour chacune des clauses en appelant un P-oracle. Ainsi, $\mathbf{CE}^{dec}$ se retrouve dans P^P pour les P2PIS KROM, c-à-d. dans Δ_1^P, donc dans P.

Enfin, si on considère les P2PIS Horn (resp. Horn inverse), le langage (cible) d'un pair donné contient au plus un nombre de clauses de Horn (resp. Horn inverse) qui est exponentiel dans la taille de son alphabet : le nombre des clauses purement négatives (resp. positives) est déjà $2^{\#\mathcal{A}(\mathcal{P})}$ pour un pair donné $\mathcal{P}$ (elles sont obtenues en faisant la disjonction de soit $\neg V$ (resp. V) ou $\Box$ pour chaque variable V de $\mathcal{A}(\mathcal{P})$). De plus, il a été montré que décider si une clause est une conséquence d'une théorie de clauses de Horn (resp. Horn inverse) est dans P (par exemple [38]). L'appartenance à Π_1^P découle ici du fait que le problème complémentaire est dans Σ_1^{PP}, c-à-d. Σ_1^P : il suffit de deviner une clause dans le langage cible de $\mathcal{P}$ et de tester, en appelant un P-oracle, que la Propriété Ψ de la Définition 4 n'est pas vérifiée (c-à-d. $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \models c$ et $\mathcal{T}(\mathcal{P}) \not\models c$). $\Box$

3.1.2 Problème fonctionnel

Notre problème fonctionnel, $\mathbf{CE}^{fun}$, 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 conservativité de ce pair.

Définition 6 (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 $\mathcal{T}(\mathcal{P}) \not\models c$.*

En se basant sur la définition ci-dessus, notre problème fonctionnel est le suivant.

Définition 7 (Problème fonctionnel : $\mathbf{CE}^{fun}$)
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 $\mathcal{S}$?*

D'un point de vue complexité espace/temps, répondre à $\mathbf{CE}^{fun}$ est en général plus difficile que décider $\mathbf{CE}^{dec}$. Ceci est du à une consommation supplémentaire exponentielle en espace afin

d'engendrer les témoins de non conservativité.

On peut facilement voir que résoudre $\mathbf{CE}^{fun}$ est au moins aussi difficile que décider $\mathbf{CE}^{dec}$ puisque le premier fournit un moyen de résoudre le deuxième. 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.

Il découle alors du théorème 2 et de la hiérarchie des classes de complexité $P \subseteq \Pi_1^P \subseteq \Delta_2^P \subseteq \Pi_2^P \subseteq PSPACE$ que résoudre $\mathbf{CE}^{fun}$ exige au moins un espace polynomial dans les P2PIS clausaux, $\mathbf{K}_{\geq \mathbf{3}}$, Horn (inverse), ou KROM ; et exige au moins un temps polynomial dans les P2PIS KROM ($\mathbf{CE}^{dec}$ est dans P), et peut exiger un temps exponentiel dans les P2PIS $\mathbf{K}_{\geq \mathbf{3}}$ ou Horn (inverse) ($\mathbf{CE}^{dec}$ est dans Δ_2^P, respectivement dans Π_1^P), et exige un temps exponentiel dans les P2PIS clausaux ($\mathbf{CE}^{dec}$ est complet pour Π_2^P).

En fait, répondre à $\mathbf{CE}^{fun}$ est aussi difficile que décider $\mathbf{CE}^{dec}$ dans les P2PIS $\mathbf{K}_{\geq \mathbf{3}}$ ou KROM, et strictement plus difficile dans les P2PIS clausaux ou Horn (inverse).

En effet, considérons un P2PIS insatisfiable $\mathcal{S}$ dont un pair $\mathcal{P}$ est tel que $\mathcal{T}(\mathcal{P}) = \{\}$ (c-à-d. *true*) et $Tgt(\mathcal{L}(\mathcal{P})) = \mathcal{L}(\mathcal{P})$. Évidemment, toutes les clauses dans le langage de $\mathcal{P}$ sont des témoins de l'extension non conservative de $\mathcal{P}$ dans $\mathcal{S}$. Ce nombre est exponentiel dans la taille de $\mathcal{A}(\mathcal{P})$ pour les clauses sans restriction (plus précisément $3^{\#\mathcal{A}(\mathcal{P})}$) et pour les clauses Horn (inverse) (entre $2^{\#\mathcal{A}(\mathcal{P})}$, le nombre des clauses purement négatives (resp. positives) sans redondance de littéraux, et $3^{\#\mathcal{A}(\mathcal{P})}$, la taille de $\mathcal{L}(\mathcal{P})$), alors que c'est un polynôme de degré $\mathbf{K}_{\geq \mathbf{3}}$ dans la taille de $\mathcal{A}(\mathcal{P})$ pour les clauses $\mathbf{K}_{\geq \mathbf{3}}$ et que ce nombre est quadratique dans la taille de $\mathcal{A}(\mathcal{P})$ pour les clauses de KROM. Par conséquent, dans les P2PIS $\mathbf{K}_{\geq \mathbf{3}}$ ou KROM, un algorithme peut être facilement conçu pour résoudre $\mathbf{CE}^{fun}$ tel que sa complexité coïncide avec celle de décider $\mathbf{CE}^{dec}$. Il suffit de générer, en temps et espace polynomiaux, toutes les clauses dans le langage d'un pair, et puis retourner toutes celles qui ne satisfont pas la Propriété Ψ de la Définition 4. Elles peuvent être exhibées par un test de conséquences clausales qui est dans P pour les clauses de KROM et dans Π_1^P-complet pour les clauses $\mathbf{K}_{\geq \mathbf{3}}$. En revanche, tout algorithme résolvant $\mathbf{CE}^{fun}$ dans les P2PIS clausaux ou Horn (inverse) doit, dans le pire des cas, générer et retourner un nombre exponentiel de témoins.

Observons que décider $\mathbf{CE}^{dec}$ ne souffre pas d'un tel besoin d'espace exponentiel additionnel. La raison est que la mémoire peut être réutilisée en générant *une à une* un nombre de clauses possiblement exponentiel dans le langage (cible) d'un pair – en ordonnant les littéraux du pair – jusqu'à trouver un témoin à la non conservativité (s'il y en a).

Exemple. 8 (Extension conservative) Le P2PIS de la Figure 2.1 (*cf.* page 16) n'est ni une extension conservative de $\mathcal{P}_1$, ni une extension conservative de $\mathcal{P}_2$. (Il est toutefois une extension conservative de $\mathcal{P}_3$.)

En effet, $\neg L_1$ est une conséquence de $\{\neg L_1 \vee Cher_2, \neg Cher_2 \vee Fac_3, \neg Fac_3 \vee MFac_3, \neg MFac_3 \vee \neg L_1\} \subset \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ avec $\mathcal{T}(\mathcal{P}_1) \not\models \neg L_1$ et $\neg Etu_2 \vee Col_2 \vee Lyc_2 \vee Univ_2$ est une conséquence de $\{\neg ET_2 \vee Prim_2 \vee Col_2 \vee Lyc_2 \vee Univ_2, \neg Prim_2 \vee Dip_3 \vee PhDS_3, \neg Dip_3 \vee L_1 \vee M_1, \neg L_1 \vee Cher_2, \neg Cher_2 \vee Fac_3, \neg Fac_3 \vee MFac_3, \neg MFac_3 \vee \neg L_1, \neg M_1 \vee Col_2, \neg PhDS_3 \vee I_1, \neg I_1 \vee Univ_2\} \subset \mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S})$ avec $\mathcal{T}(\mathcal{P}_2) \not\models \neg Etu_2 \vee Col_2 \vee Lyc_2 \vee Univ_2$.

Il découle que dans le P2PIS, $\mathcal{P}_1$ ne propose pas de cours de langues (et donc il propose seulement des cours de maths et d'informatique) et $\mathcal{P}_2$ ne possède pas des étudiants en école primaire, que ces pairs le veuillent ou non.

Notons que le témoin $\neg L_1$ est *incomparable* (par rapport à $\models$) avec les connaissances de $\mathcal{T}(\mathcal{P}_1)$, alors qu'une clause de $\mathcal{T}(\mathcal{P}_2)$ est une conséquence du témoin $\neg Etu_2 \vee Col_2 \vee Lyc_2 \vee Univ_2$. ■

3.1.3 Fondements de l'algorithme CECA

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 d'*impliqués premiers d'une théorie propositionnelle* (*cf.* Chapitre 1).

L'ensemble des impliqués premiers d'une théorie clausale peut être vu comme le plus petit ensemble représentatif de toutes les conséquences de cette théorie. Ainsi, trouver l'ensemble des impliqués premiers d'un P2PIS qui sont dans le langage cible d'un pair donné est *suffisant* pour résoudre notre problème de décision et notre problème fonctionnel, pour ce pair. Fort heureusement, d'un point de vue "optimisation", il s'avère que trouver *tous* ces impliqués premiers n'est *pas nécessaire* : le nombre d'impliqués premiers d'une théorie clausale peut être exponentiel dans la taille de cette théorie (Propriété 1 du chapitre 1).

La Proposition 5 établit en effet que nous avons *seulement* à considérer les impliqués premiers dans le langage d'un pair qui sont propres à un mapping de ce pair. La preuve consiste à montrer que tout autre impliqué premier est sans intérêt pour la notion 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 4), soit il n'est pas exprimé dans le langage (cible) de ce pair (c-à-d. il n'a alors rien à voir avec la Propriété Ψ de la Définition 4).

Proposition 5 *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 c est un impliqué propre d'un mapping de $\mathcal{M}(\mathcal{P})$.*

Preuve. Notons $support(c)$ un sous ensemble minimal d'une théorie utilisé duquel découle c, c-à-d. dans lequel retirer une clause ne permet plus d'impliquer logiquement c.

Si $c \in \mathbf{PI}(\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}))$ et $c \in Tgt(\mathcal{L}(\mathcal{P}))$, alors il existe des clauses de $\mathcal{T}(\mathcal{P}) \cup \mathcal{M}(\mathcal{P})$ dans $support(c)$. Puisque $\mathcal{T}(\mathcal{P}) \not\models c$, il existe alors dans $support(c)$ des clauses qui ne sont pas dans $\mathcal{T}(\mathcal{P})$. Si on suppose que $support(c)$ ne contient pas de mappings de $\mathcal{M}(\mathcal{P})$, $support(c)$ est alors constitué de deux ensembles indépendants de clauses : un ensemble pour les clauses de $\mathcal{T}(\mathcal{P})$ et un autres pour celles qui ne sont pas dans $\mathcal{T}(\mathcal{P}) \cup \mathcal{M}(\mathcal{P})$, ces deux ensembles n'utilisant pas le même alphabet. Ainsi, toute conséquence de $support(c)$ découle soit d'un ensemble, soit de l'autre ensemble, ce qui est contradictoire avec le fait que $support(c)$ est minimal. Par conséquent, $support(c)$ contient nécessairement une clause $m \in \mathcal{M}(\mathcal{P})$ et c est un impliqué *propre* de m. □

La Proposition 6 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 linéaires dont les top clauses sont les mappings de ce pair.

Proposition 6 *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 $\mathcal{T}(\mathcal{S}) \cup \mathcal{M}(\mathcal{S}) \vdash^m_{LR} c$.*

Preuve. Considérons n'importe quel support *minimal* de clauses $support(c)$ du P2PIS du quel découle c. Par la Proposition 5, $support(c)$ contient au moins un mapping du pair $\mathcal{P}$, pour lequel c est un impliqué propre. Par la Propriété 2 (troisième point), c est dérivable linéairement à partir de ce mapping dans $support(c)$ et donc dans le P2PIS. □

Remarquons que la Proposition 5 et la Proposition 6 sont vraies *uniquement* pour des P2PIS *satisfiables*. La raison à cela est qu'un P2PIS insatisfiable a un unique impliqué premier, la clause vide □, qui appartient au langage cible de tous les pairs. Par conséquent, ces propositions seraient vérifiées si □ pouvait être dérivée (par déduction linéaire) à partir d'un mapping de chaque pair, ce qui n'est – bien évidemment – pas le cas (par exemple 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 3.2, une stratégie (décentralisée) auto-guérissante permettant de rendre satisfiable un P2PIS insatisfiable.

À partir des propriétés de la déduction (linéaire) et de la Proposition 6, nous aboutissons finalement au théorème suivant qui reformule la notion d'extension conservative d'un pair en termes de calcul de conséquences.

Theorème 3 (Test de l'extension conservative) *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 $m \in \mathcal{M}(\mathcal{P})$, 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 calculer $\mathcal{T}(\mathcal{P}) \cup \{\neg c\} \vdash_R \square$). On notera qu'il existe déjà plusieurs solutions pour la déduction/réfutation dans une théorie de LP (par exemple [23, 55]), et que nous avons proposé l'algorithme de déduction linéaire décentralisée DeCA$_{LR}$ dans le Chapitre 2. Nous allons donc faire appel à cette première contribution de la thèse, en considérant ici les optimisations données [entre crochets] dans DeCA$_{LR}$.

3.1.4 L'algorithme CECA

Pour résoudre notre problème de décision ($\mathbf{CE}^{dec}$) et notre problème fonctionnel ($\mathbf{CE}^{fun}$), nous proposons l'algorithme CECA qui utilise à son tour l'algorithme DeCA$_{LR}$ présenté dans le Chapitre 2. Nous décrivons CECA et nous établissons ses propriétés et sa complexité communicationnelle et en terme d'espace et de temps.

Description

En suivant le Théorème 3, CECA^k (CECA sur $\mathcal{P}_k$) considère chaque mapping de $\mathcal{M}(\mathcal{P}_k)$ (ligne 2) à partir duquel il calcule grâce à DeCA$_{LR}$ des impliqués du P2PIS *dans le langage cible de $\mathcal{P}_k$*, y compris *tous* les impliqués premiers et propres à ce mapping (ligne 3). Ici, le point subtil est que CECA recoure à l'opérateur $\lozenge$, en suivant la Proposition 2, 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 Non sitôt qu'il trouve un impliqué témoignant de la non conservativité du pair $\mathcal{P}_k$, ou en répondant Oui 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 $temoins = temoins \cup \{(l_k^1 \vee \cdots \vee l_k^n, l_1 \vee \cdots \vee l_p)\}$. Ceci est utile pour un pair dans le cas où il veut détecter/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 $dp = vrai$: Oui si $\mathcal{S}$ est une extension conservative de $\mathcal{P}_k$, Non 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 $l_1 \vee \cdots \vee l_p \in \mathcal{M}(\mathcal{P}_k)$
(3) **foreach** $l_k^1 \vee \cdots \vee l_k^n \in \oslash_{i=1}^{p} \text{DeCA}_{LR}{}^i(l_i, \emptyset, k)$ tel que $l_k^1 \vee \cdots \vee l_k^n \in Tgt(\mathcal{L}(\mathcal{P}_k))$
(4) **if** $\mathcal{T}(\mathcal{P}_k) \cup \{\bar{l}_k^1, \ldots, \bar{l}_k^n\} \not\vdash_R \square$, où $\bar{l}_k^i$ est le complément de l_k^i pour $i \in [1..n]$
(5) **if** dp
(6) **return** Non
(7) **else**
(8) $temoins = temoins \cup \{l_k^1 \vee \cdots \vee l_k^n\}$
(9) **if** dp
(10) **return** Oui
(11)**else**
(12) **return** $temoins$

Exemple. 9 (Exécution de CECA**)** Nous avons mentionné précédemment que le P2PIS de la Figure 2.1 (page 16) 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 $\neg L_1 \vee Cher_2$ ou le mapping $\neg MFac_3 \vee \neg L_1$ sont traités : $\text{DeCA}_{LR}{}^1(\neg L_1, \emptyset, 1) \oslash \text{DeCA}_{LR}{}^2(Cher_2, \emptyset, 1)$ et $\text{DeCA}_{LR}{}^1(\neg L_1, \emptyset, 1) \oslash \text{DeCA}_{LR}{}^3(\neg MFac_3, \emptyset, 1)$ produisent tous les deux $\neg L_1$, $\neg C_1 \vee M_1 \vee CS_1$, et $\neg C_1 \vee M_1 \vee CS_1 \vee \neg L_1$ comme témoins à la non-conservativité de $\mathcal{P}_1$. Par conséquent, CECA$^1(vrai)$ retourne Non et CECA$^1(faux)$ retourne les témoins mentionnés ci-dessus.

La non conservativité de $\mathcal{P}_2$ est exhibée quand le mapping $\neg Prim_2 \vee Dip_3 \vee PhDS_3$ est traité. En effet, $\text{DeCA}_{LR}{}^2(\neg Prim_2, \emptyset, 2) \oslash \text{DeCA}_{LR}{}^3(Dip_3, \emptyset, 2) \oslash \text{DeCA}_{LR}{}^3(PhDS_3, \emptyset, 2)$ produit $\neg Prim_2 \vee Col_2 \vee Univ_2$, $\neg Etu_2 \vee Col_2 \vee Lyc_2 \vee Univ_2$, et $\neg Ap_2 \vee Cher_2 \vee Col_2 \vee Lyc_2 \vee Univ_2$ comme témoins à la non-conservativité de $\mathcal{P}_2$. Par conséquent, CECA$^2(vrai)$ retourne Non et CECA$^2(faux)$ retourne les témoins mentionnés ci-dessus.

Enfin, le P2PIS est une extension conservative de $\mathcal{P}_3$ puisqu'il n'y a pas moyen de dériver de témoins à la non-conservativité de $\mathcal{P}_3$ à partir de ses mappings. Ainsi, CECA$^3(vrai)$ retourne Oui et CECA$^3(faux)$ retourne un ensemble vide de témoins. ■

Propriétés

Le Théorème 4 établit que CECA est une procédure effective pour répondre à nos deux problèmes.

Theorème 4 *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}_k$.*
- CECA$^k(true)$ *et* CECA$^k(false)$ *terminent.*
- CECA$^k(true)$ *retourne* Oui *ssi $\mathcal{S}$ est une extension conservative de $\mathcal{P}_k$.*
- CECA$^k(false)$ *retourne un ensemble de témoins de la non conservativité de $\mathcal{P}_k$, dont tous les impliqués premiers découlant nécessairement des mappings de $\mathcal{P}_k$.*

Preuve. Ce théorème découle directement du Théorème 3, du Théorème 1 et de la Proposition 2 (pour la ligne 3). □

Complexité

Nous arrivons maintenant à l'étude des complexités de CECA en termes de communication, d'espace et de temps. Comme pour DeCA$_{LR}$, nous nous intéressons au nombre de messages échangés (pour la communication), de clauses à stocker simultanément (pour l'espace) et au nombre de clauses à générer (pour le temps) afin de calculer CECA(dp). Dans ce qui suit, nous faisons deux suppositions sur l'implémentation de CECA. La première est le *caching* consistant ici à faire un appel *unique* à DeCA$_{LR}$ par littéral distinct d'un pair distant apparaissant dans $\mathcal{M}(\mathcal{P}_k)$ au lieu de faire un appel récursif pour chaque occurrence d'un tel littéral pendant le traitement des itérations sur les mappings aux lignes 2 et 3. La deuxième supposition est que CECA peut aussi utiliser les stratégies *en profondeur d'abord* et *en parallèle* pour déclencher les appels à DeCA$_{LR}$ (*cf.* Chapitre précédent). Pour simplifier, nous supposons que CECA et DeCA$_{LR}$ suivent tous les deux soit la stratégie en profondeur d'abord ou la stratégie en parallèle.

Complexité communicationnelle La Proposition 7 fournit une borne supérieure de la complexité de communication en termes du nombre de messages échangés au sein du P2PIS pour calculer CECA$^k(dp)$.

Proposition 7 *Soient $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..m}$ un P2PIS et $\mathcal{P}_k$ un pair de $\mathcal{S}$. Soit n le nombre maximal de littéraux distincts de pairs distants apparaissant dans l'ensemble des mappings de n'importe quel pair. Une borne supérieure du nombre de messages échangés au sein de $\mathcal{S}$ afin de calculer* CECA$^k(dp)$ *est $n * 2 * \Sigma_{i=1}^{\#\mathcal{A}(\mathcal{S})+1} n^i$.*

La Proposition 7 découle directement de l'hypothèse de caching et la Proposition 3 du Chapitre 2.

Complexité de CECA **en espace et en temps** La Proposition 8 établit les complexités de CECA en termes d'espace et de temps dans le pire des cas.

On peut remarquer ici que CECA résout $\mathbf{CE}^{dec}$ et $\mathbf{CE}^{fun}$ avec les mêmes complexités en terme d'espace et de temps. Pourtant, nous avons montré d'un point de vue théorique que $\mathbf{CE}^{fun}$ est strictement plus difficile que $\mathbf{CE}^{dec}$ dans le cas général (*cf.* Section 3.1). La raison est que nous ne proposons pas un algorithme spécifique et optimisé dédié à $\mathbf{CE}^{dec}$. Nous utilisons un algorithme consacré à $\mathbf{CE}^{fun}$ afin de résoudre $\mathbf{CE}^{dec}$, puisque pour un pair donné, un ensemble

vide de témoins montre la conservativité et un ensemble non vide montre la non conservativité. Ce choix n'est – certes – pas optimal d'un point de vue théorique, mais fournir un algorithme pour $\mathbf{CE}^{dec}$ n'a pas/peu d'utilité pratique concernant la (non) conservativité d'un pair. Toutefois, l'étude théorique de $\mathbf{CE}^{dec}$ est intéressante pour trouver une borne inférieure des complexités de $\mathbf{CE}^{fun}$ en termes d'espace et de temps.

La Proposition 8 établit que résoudre par CECA $\mathbf{CE}^{dec}$ et $\mathbf{CE}^{fun}$ exigent dans le pire des cas un espace exponentiel dans la taille d'un pair et un temps exponentiel dans la taille du P2PIS quand les appels (récursifs) à DeCA$_{LR}$ sont déclenchés avec la stratégie en profondeur d'abord ; et un espace exponentiel dans la taille du P2PIS et un temps exponentiel dans la taille d'un pair quand les appels (récursifs) à DeCA$_{LR}$ sont déclenchés avec la stratégie en parallèle. Cela veut dire que les complexités en termes d'espace et de temps de $\mathbf{CE}^{fun}$ (voir la Section 3.1) et de CECA coïncident avec la stratégie en profondeur d'abord, alors que CECA n'est pas optimal en terme de consommation d'espace pour $\mathbf{CE}^{fun}$ avec la stratégie en parallèle.

Proposition 8 *Soit $\mathcal{S}$ un P2PIS satisfiable dont un pair est $\mathcal{P}_k$. Soit a la taille maximale de l'alphabet de n'importe quel pair : $a = max(\mathcal{A}(\mathcal{P}_1), \ldots, \mathcal{A}(\mathcal{P}_m))$. Soit n le nombre maximal de littéraux distincts de pairs distants apparaissant dans l'ensemble de mappings de n'importe quel pair.*
(i) Une borne supérieure de l'espace nécessaire pour calculer CECA$^k(true)$ *ou* CECA$^k(false)$ *est :*

- *dans $O(\#\mathcal{A}(\mathcal{S}) * n^2 * 3^{a+n+1})$ clauses quand* CECA *et* DeCA$_{LR}$ *suivent la stratégie en profondeur d'abord ;*
- *dans $O(n^{\#\mathcal{A}(\mathcal{S})+4} * 3^{a+n+1})$ clauses quand* CECA *et* DeCA$_{LR}$ *suivent la stratégie en parallèle.*

(ii) Une borne supérieure du temps nécessaire pour calculer CECA$^k(true)$ *ou* CECA$^k(false)$ *est :*

- *dans $O(n^{\#\mathcal{A}(\mathcal{S})+3} * 3^{a*(a+n+1)+n})$ clauses à générer quand* CECA *et* DeCA$_{LR}$ *suivent la stratégie en profondeur d'abord ;*
- *dans $O(\#\mathcal{A}(\mathcal{S}) * 3^{a*(a+n+1)+n})$ clauses à générer quand* CECA *et* DeCA$_{LR}$ *suivent la stratégie en parallèle.*

Preuve. Considérons d'abord l'espace nécessaire pour exécuter CECA($true$) or CECA($false$).

CECA a besoin de stocker les clauses qui sont gérées par $\mathcal{P}_k$, c-à-d. $\mathcal{T}(\mathcal{P}_k) \cup \mathcal{M}(\mathcal{P}_k)$. Une borne maximale de ces clauses est de $[\alpha]$ 3^{a+n} (voir les complexités de DeCA$_{LR}$ en termes d'espace et de temps). Ensuite, CECA a besoin de calculer les appels à DeCA$_{LR}$ et stocker leurs résultats. Le nombre d'appels est borné par n puisque nous considérons le caching. Le calcul de tous les appels à DeCA$_{LR}$ exigent au plus un espace en $[\beta]$ $O(\#\mathcal{A}(\mathcal{S})^2 * n * 3^{a+n+1})$ clauses (voir Proposition 4 avec $q = 1$ dans notre cas) quand CECA et DeCA$_{LR}$ utilisent tous les deux la stratégie en profondeur d'abord ; et un espace $[\gamma]$ n multiplié par un nombre en $O(n^{\#\mathcal{A}(\mathcal{S})+3} * 3^{a+n+1})$ clauses (voir Proposition 4 avec $q = 1$ dans notre cas) quand CECA et DeCA$_{LR}$ utilisent tous les deux la stratégie en parallèle. Le stockage des résultats de tous les appels à DeCA$_{LR}$ exige au plus $[\delta]$ $n * 3^a$ clauses, puisque ces appels retournent des clauses dans le langage de $\mathcal{P}_k$ seulement. Enfin, CECA a besoin de calculer à partir des résultats des appels à DeCA$_{LR}$ (avec $\Diamond$) les impliqués du P2PIS, et de réfuter chacun d'eux. Ces impliqués étant dans le langage de $\mathcal{P}_k$, nous avons au plus $[\epsilon]$ 3^a impliqués à stocker et réfuter. Chaque réfutation utilise des clauses constituées de variabes de $\mathcal{A}(\mathcal{P}_k)$ seulement, et donc n'importe quelle réfutation peut produire au plus $[\zeta]$ 3^a résolvants (réduits). De $[\alpha]$, $[\beta]$, $[\delta]$, $[\epsilon]$, et $[\zeta]$ d'une part, et de $[\alpha]$, $[\gamma]$, $[\delta]$, $[\epsilon]$, et $[\zeta]$ d'autre part, nous obtenons l'élément (i) de la Proposition 8.

Considérons maintenant la complexité de CECA en termes de temps.

CECA a besoin de déclencher des appels à DECA$_{LR}$ (bornés par n puisque nous considérons le caching), et donc il calcule des impliqués qui découlent des mappings aux lignes 2 et 3. Il a besoin au plus d'un temps dans $[\alpha]$ n multiplié par un nombre en $O(n^{\#\mathcal{A}(\mathcal{S})+2} * 3^{a*n+1})$ clauses à générer (voir Proposition 4 avec $q = 1$ dans notre cas) quand CECA et DECA$_{LR}$ utilisent tous les deux la stratégie en profondeur d'abord ; et en $[\beta]$ $O(\#\mathcal{A}(\mathcal{S}) * 3^{a*n+1})$ clauses à générer (voir Proposition 4 avec $q = 1$ dans notre cas) quand CECA et DECA$_{LR}$ utilisent tous les deux la stratégie en parallèle. Ensuite, et pour chaque mapping de $\mathcal{M}(\mathcal{P}_k)$, CECA doit construire les impliqués du mapping avec $\oslash$. Puisqu'un mapping implique des variables de $\mathcal{A}(\mathcal{P}_k)$ (borné par a) et des variables des pairs distants (borné par n), le nombre de mappings est borné par 3^{a+n} et le nombre de littéraux dans un mapping est borné par $a + n$. De plus, n'importe quel appel à DECA$_{LR}$ fournit au plus 3^a clauses (voir la complexité de CECA en termes d'espace). Par conséquent, le calcul des impliqués à réfuter est borné par $[\gamma]$ $3^{a+n} * 3^{a*(a+n)}$. Les impliqués à réfuter étant dans le langage de $\mathcal{P}_k$, leur nombre par mapping est borné par 3^a. Comme mentionné ci-dessus, une réfutation peut produire au plus 3^a résolvants (réduits) (voir la complexité de DECA$_{LR}$ en termes d'espace). Il découle que le temps total de réfutation est $[\delta]$ $3^{a+n} * 3^{2*a}$. De $[\alpha]$, $[\gamma]$, et $[\delta]$, respectivement de $[\beta]$, $[\gamma]$, et $[\delta]$, nous obtenons l'élément (ii) de la Proposition 8. □

3.2 Extension conservative et P2PIS insatisfiables

Nous avons expliqué dans la Section 3.1.3 que la Proposition 5, la Proposition 6, et le Théorème 3 et le Théorème 4, sont valides *uniquement* pour des P2PIS *satisfiables*. Nous montrons ici que nous avons les outils nécessaires pour élaborer des stratégies décentralisées auto-guérissantes de sorte à ce que nous puissions supposer/rendre tout P2PIS 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 □ (par exemple 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 pour le calcul de conséquences, l'insatisfiabilité d'un P2PIS peut être prouvée par des déductions linéaires dans lesquelles au moins un mapping est impliqué : □ est nécessairement un impliqué (premier) propre d'un mapping. Par la complétude de la déduction linéaire pour le calcul des impliqués premiers propres de la top clause, 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 $l_1 \vee \cdots \vee l_p$ si $\square \in$ DECA$_{LR}{}^1(l_1, \emptyset, k) \oslash \cdots \oslash$ DECA$_{LR}{}^p(l_p, \emptyset, k)$ (Proposition 2 et Théorème 1).

Si c'est le cas, il peut réparer – certes naïvement – en retirant le mapping $l_1 \vee \cdots \vee l_p$ de $\mathcal{M}(\mathcal{P}_1), \ldots, \mathcal{M}(\mathcal{P}_p)$. Faire cela revient exactement à retirer le mapping $l_1 \vee \cdots \vee l_p$ de $\mathcal{M}(\mathcal{P}_1), \ldots,$ $\mathcal{M}(\mathcal{P}_p)$ à chaque fois que $(\square, l_1 \vee \cdots \vee l_p) \in CECA^k(false)$, lorsque CECA retourne les témoins à la non-conservativité ainsi que leurs causes comme expliqué dans la Section 3.1.4.

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 exemple 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" ou "pertinents" sont supprimés), mais ceci sort du cadre de cette thèse.

3.3 Travaux connexes

À notre connaissance, cette thèse est la première à traiter la notion d'extension conservative dans un cadre décentralisé. En centralisé en revanche, le test d'extension conservative a été récemment le sujet d'un grand nombre de travaux de recherche en logiques de description.

Le test d'extension conservative a été principalement étudié dans le contexte de l'évolution d'ontologie, souvent exprimées en logiques de description. Durant leur cycle de vie, les ontologies sont souvent soumises à plusieurs types de modifications tel que le raffinement, la fusion avec une autre ontologie, la modularisation, etc. . Ces modifications peuvent parfois avoir des conséquences inattendues ou indésirables sur l'ontologie originale si celle-ci est utilisée par une large communauté. Un outil de raisonnement permettant de prévoir l'impact d'une évolution d'ontologie devient dans ce cas très utile. Plusieurs travaux de la littérature tentent d'exhiber de tels mécanismes en se basant sur le test d'extension conservative.

[13] était le premier à exhiber le lien entre le test d'extension conservative et l'évolution d'ontologie. Il s'intéresse particulièrement au raffinement. Une ontologie O' est un raffinement d'une ontologie O si elle contient plus de détails sur une partie qui n'était pas encore suffisamment décrite dans O. En suivant le principe que ces détails ne doivent pas avoir de conséquences indésirables sur la sémantique de l'ontologie originale, O' est un raffinement de O si et seulement si O' est une extension conservative de O. Dans ce cas, O est considérée comme une abstraction de O'.

[41, 53] ont étudié la notion d'extension conservative dans les logiques de description de la famille $\mathcal{ALC}$.
[41] prouve que, dans $\mathcal{ALC}$, le problème de décider si la nouvelle ontologie T_2 est une extension conservative de l'ancienne T_1 est 2EXPTIME-complet et que si elle n'est pas conservative, il existe un concept témoin C de taille 3-exponentiel dans la taille de $T_1 \cup T_2$.
[53] part de ce résultat et considère le même problème pour des logiques plus expressives que $\mathcal{ALC}$ telles que $\mathcal{ALCQI}$ et $\mathcal{ALCQIO}$. Il prouve que décider de l'extension conservative dans $\mathcal{ALCQI}$ est 2EXPTIME-complet et donc pas plus difficile que $\mathcal{ALC}$ alors que le même problème est indécidable pour $\mathcal{ALCQIO}$. Puisque $\mathcal{ALCQIO}$ est un fragment de OWL-DL, le problème d'extension conservative est alors indécidable dans OWL-DL. $\mathcal{ALCQI}$ peut être défini donc comme un fragment significatif de OWL-DL pour lequel le problème d'extension conservative reste décidable. [41, 53] proposent donc des algorithmes de décision de l'extension conservative pour $\mathcal{ALC}$ et $\mathcal{ALCQI}$.

[54] considère des logiques de description de la famille $\mathcal{EL}$ pour lesquelles décider si une ontologie est une extension conservative d'une autre est EXPTIME-complet. [54] propose un algorithme de décision pour le test d'extension conservative.

Enfin, [52] considère deux membres de la famille de DL-LITE DL-LITE$_{bool}$ et DL-LITE$_{horn}$ et prouve que le problème consistant à décider si une ontologie $\mathcal{T}_1 \cup \mathcal{T}_2$ est une extension conservative de $\mathcal{T}_1$ est coNP-complet pour DL-LITE$_{horn}$ et Π_2^P-complet pour DL-LITE$_{bool}$.

Si les travaux précédents traient de l'extension, de la fusion et du raffinement d'ontologies, [44, 42, 43] s'intéressent plutôt à la notion de modularité, l'équivalent de la notion de modularité en Ingénierie du Logiciel. En effet, la modularité consiste à décomposer une ontologie en plusieurs modules ce qui facilite la compréhension, la vérification, l'extension, et la réutilisation d'une ontologie. Un module appartenant à une ontologie $\mathcal{T}$ et décrivant une partie indépendante du domaine d'application peut être défini comme une sous-ontologie $\mathcal{T}'$ de $\mathcal{T}$ tel que $\mathcal{T}$ est une extension conservative de $\mathcal{T}'$ par rapport aux noms de concepts et de rôles appartenant à $\mathcal{T}'$.

De plus, selon [44], les modules doivent être petits, indépendants et intelligibles afin de faciliter la réutilisation. Il propose donc un algorithme qui identifie et extrait des modules à partir d'une ontologie $\mathcal{SHOIN}$. [43] généralise [44] et propose un cadre logique pour la modularité des ontologies $\mathcal{SHIQ}$ et $\mathcal{SHOIQ}$. Dans [42], le problème est posé différemment dans OWL : étant donné un ensemble de termes E, on cherche un module qui garantit de capter tous les axiomes appropriés à la signification de E et que ce module soit minimal. Ce problème est prouvé indécidable dans OWL-DL même pour des fragments restreints tel que $\mathcal{ALCO}$ et donc on ne peut pas déterminer si un sous-ensemble d'une ontologie est un module pour un vocabulaire donné. Deux approximations (définitions alternatives de modules pour un vocabulaire) sont alors proposées : une approximation sémantique et une syntaxique qui sont calculées en temps polynomial. Elles fournissent les garanties suggérées sauf qu'elles sont assez strictes et peuvent donc fournir des modules plus grands.

Deuxième partie

Raisonnement décentralisé en DL-Lite [2]

2. Les contributions de cette partie résultent d'une collaboration avec Marie-Christine Rousset (PR, Univ. grenoble). Elles ont été publiées à l'*International Joint Conference on Articial Intelligence* IJCAI en 2009 [4] et au *congrès francophone de Reconnaissance de Formes et Intelligence Artificielle* RFIA en 2010 [5].

Chapitre 4

Préliminaires : DL-LITE$_\mathcal{R}$

Les *ontologies* sont au cœur du Web Sémantique. Elles fournissent une vue conceptuelle des données et services mis à disposition sur le Web, dans le but de faciliter leur manipulation.

Le langage d'ontologies recommandé par le W3C, OWL, se fonde sur les *logiques de description* (LD) [16]. Celles-ci couvrent un large spectre de langages pour lesquels raisonner est (la plupart du temps) décidable avec une complexité variant en fonction des constructeurs autorisés.

Une base de connaissances en LD est constituée d'une partie intentionnelle, la *Tbox*, et d'une partie assertionnelle, la *Abox* : la Tbox définit l'ontologie en termes de laquelle sont décrites les données dans la Abox.

Répondre à des requêtes conjonctives (c-à-d. sélection-projection-jointure) sur des ontologies est un problème central pour la mise en oeuvre du Web Sémantique. Contrairement à de nombreux autres problèmes de raisonnement, répondre à une requête n'est pas réductible au problème de (in)satisfiabilité.

La famille DL-LITE [25] a été spécialement définie la gestion de données à grande échelle. Répondre à une requête est effectué par une approche à base de reformulation de requête qui (1) calcule les requêtes conjonctives les plus générales qui, avec les axiomes de la Tbox, impliquent logiquement la requête initiale et (2) évalue chacune de ces reformulations de requête sur la Abox vue comme une base de données relationnelles. Une telle approche a pour intérêt de rendre possible l'utilisation d'un moteur de requêtes SQL dans la seconde étape, et donc de bénéficier des stratégies d'optimisation implantées dans les systèmes de gestion de bases de données. L'étape de reformulation (1) est nécessaire afin de garantir la complétude des réponses. Le point important est qu'elle est indépendante des données (c-à-d. de la Abox).

Nous nous intéressons ici au dialecte DL-LITE$_\mathcal{R}$ qui est un fragment de OWL [3] et de son évolution OWL2 [4]. Notamment, le profile QL de OWL2 se fonde sur DL-LITE$_\mathcal{R}$. De plus, DL-LITE$_\mathcal{R}$ étend RDFS [5] (le standard du W3C pour écrire des ontologies simples) avec la possibilité de déclarer, par exemple, des relations disjointes ou d'exprimer des rôles inverses.

Nous présentons la logique DL-LITE$_\mathcal{R}$ dans la Section 4.1, ainsi que les deux problèmes de gestion de données – et les algorithmes pour les résoudre – qui vont nous intéresser par la suite : la vérification de la consistance dans la Section 4.2 et répondre à des requêtes dans la Section 4.3. Ce chapitre est fondé sur [25].

3. http ://www.w3.org/2004/OWL/
4. http ://www.w3.org/TR/owl2-overview/
5. http ://www.w3.org/TR/rdf-schema/

4.1 DL-LITE$_\mathcal{R}$ et gestion de données

Comme la majorité des logiques de description, DL-LITE$_\mathcal{R}$ n'utilise que des relations unaires, les *concepts*, et des relations binaires, les *rôles*. Les concepts et rôles en DL-LITE$_\mathcal{R}$ sont de la forme suivante :

- $B \rightarrow A \mid \exists R$,
- $C \rightarrow B \mid \neg B$,
- $R \rightarrow P \mid P^-$,
- $E \rightarrow R \mid \neg R$,

où A est un *concept atomique*, P un *rôle atomique*, et P^- l'*inverse* de P. B est un *concept basique* (c-à-d. un concept atomique A ou une *quantification existentielle non qualifiée sur un rôle basique* $\exists R$) et R un *rôle basique* (c-à-d. un rôle atomique P ou son inverse P^-). Enfin, C est un *concept général* (c-à-d. un concept basique ou sa négation) et E un *rôle général* (c-à-d. un rôle basique ou sa négation).

La sémantique des concepts et rôles de DL-LITE$_\mathcal{R}$ est donnée en termes d'*interprétations*. Une interprétation $I = (\Delta^I, .^I)$ consiste en un *domaine d'interprétation* non vide Δ^I et une *fonction d'interprétation* $.^I$ qui assigne un sous-ensemble de Δ^I à chaque concept atomique ($A^I \subseteq \Delta^I$) et une relation binaire sur Δ^I à chaque rôle atomique ($P^I \subseteq \Delta^I \times \Delta^I$). La sémantique des concepts et rôles non atomiques est définie comme suit :

- $(P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\}$,
- $(\exists R)^I = \{o_1 \mid \exists o_2 \ (o_1, o_2) \in R^I\}$,
- $(\neg B)^I = \Delta^I \backslash B^I$,
- $(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$.

Une interprétation I *satisfait* ou est un *modèle d'un concept C (resp. un rôle E)* si $C^I \neq \emptyset$ (resp. $E^I \neq \emptyset$).

Une base de connaissances (BC) de DL-LITE$_\mathcal{R}$ est constituée d'une *Tbox* représentant une modélisation conceptuelle et formelle d'un domaine d'intérêt (c-à-d. une ontologie) et d'une *Abox* (un ensemble de faits) pour représenter les données.

Une Tbox de DL-LITE$_\mathcal{R}$ est un ensemble fini d'inclusions de la forme $B \sqsubseteq C$ et/ou $R \sqsubseteq E$. Les concepts et rôles *généraux* ne sont admis qu'en partie droite des inclusions et seuls les concepts et rôles *basiques* sont autorisés en partie gauche. Les inclusions de la forme $B_1 \sqsubseteq B_2$ ou $R_1 \sqsubseteq R_2$ sont appelée des *inclusions positives (PI)*, alors que les inclusions de la forme $B_1 \sqsubseteq \neg B_2$ ou $R_1 \sqsubseteq \neg R_2$ sont appelées des *inclusions négatives (NI)*. Une PI exprime qu'un concept (ou rôle) basique en *subsume* un autre, alors qu'une NI exprime que deux concepts (ou rôles) basiques sont *disjoints*.
Une interprétation $I = (\Delta^I, .^I)$ *satisfait* ou est un *modèle d'une inclusion* $B \sqsubseteq C$ (resp. $R \sqsubseteq E$) si $B^I \subseteq C^I$ (resp. $R^I \subseteq E^I$). I *satisfait* ou est un *modèle d'une Tbox* si elle satisfait toutes les inclusions. Notamment, toute Tbox de DL-LITE$_\mathcal{R}$ est satisfiable. Une Tbox $\mathcal{T}$ *implique logiquement* une inclusion α, noté $\mathcal{T} \models \alpha$, si tout modèle de $\mathcal{T}$ est un modèle de α.

Une Abox de DL-LITE$_\mathcal{R}$ consiste en un nombre fini de faits pour des concepts et rôles *atomiques* de la forme $A(a)$ et $P(a, b)$, exprimant respectivement que a est une instance de A et que la paire de constantes (a, b) est une instance de P.
La fonction d'interprétation d'une interprétation $I = (\Delta^I, .^I)$ est étendue aux constantes en assignant à chaque constante a un objet distinct $a^I \in \Delta^I$ (c-à-d. l'hypothèse du nom unique est vérifiée). Une interprétation I *satisfait* ou est un *modèle d'un fait* $A(a)$ (resp. $P(a, b)$) si $a^I \in A^I$ (resp. $(a^I, b^I) \in P^I$). C'est un *modèle d'une Abox* si I satisfait tous les faits.

Une BC $\mathbf{K}$ est une Tbox associée à une Abox : $\langle \mathcal{T}, \mathcal{A} \rangle$. Une interprétation I *satisfait* ou est *modèle d'une BC* de $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ si elle satisfait à la fois $\mathcal{T}$ et $\mathcal{A}$. Enfin, une BC $\mathbf{K}$ *implique logiquement* une assertion (inclusion ou fait) β, noté $\mathbf{K} \models \beta$, si tout modèle de K est un modèle de β.

Requête et réponses à une requête. Nous nous intéressons ici aux (unions de) *requêtes conjonctives*, c'est-à-dire des requêtes de type selection-projection-jointure, le langage de requêtes central pour les bases de données relationnelles. Une requête conjonctive est de la forme $q(\bar{x}) = \exists \bar{y}\ conj(\bar{x}, \bar{y})$, où $conj(\bar{x}, \bar{y})$ est une conjonction d'atomes dont les variables sont *uniquement* les variables libres $\bar{x}$ et les variables existentielles $\bar{y}$, et dont les prédicats sont des concepts *atomiques* ou des rôles *atomiques* de la Tbox. L'*arité* d'une requête est le nombre de ses variables libres, par exemple 0 pour une *requête booléenne*.

Étant donnée une interprétation $I = (\Delta^I, .^I)$, la sémantique q^I d'une requête booléenne q est définie par $true$ si $[\exists \bar{y}\ conj(\emptyset, \bar{y})]^I = true$, et $false$ sinon, tandis que la sémantique q^I d'une requête q d'arité $n \geq 1$ est la relation d'arité n définie sur (Δ^I) comme suit : $q^I = \{\bar{e} \in (\Delta^I)^n \mid [\exists \bar{y}\ conj(\bar{e}, \bar{y})]^I = true\}$.

L'ensemble de réponses à une requête n-aire non booléenne q sur $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ est : $ans(q, \mathbf{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathbf{K} \models q(\bar{t})\}$, où $\mathcal{C}$ est l'ensemble des constantes apparaissant dans la Abox et $q(\bar{t})$ est la formule *close* obtenue en remplaçant les variables libres de $\bar{x}$ par les constantes de $\bar{t}$ dans la définition de la requête.
Par convention, l'ensemble des réponses à une requête booléenne est $\{()\}$, où $()$ est le tuple vide, si $\mathbf{K} \models q()$, et $\emptyset$ sinon.

Exemple. 10 (Service de cours particuliers)
Considérons la BC centralisée $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ d'un service de cours particuliers. La Figure 4.1 illustre la Tbox $\mathcal{T}$ de $\mathbf{K}$. Les concepts sont notés en majuscule et les rôle en minuscule.

Le service gère (i) des cours (C) de langues (L) ou de maths (M) qui sont disjoints ($L \sqsubseteq C$, $M \sqsubseteq C$ et $L \sqsubseteq \neg M$), (ii) des enseignants (EN) dont certains sont des mathématiciens (ENM), (iii) des étudiants (ET) dont certains sont anglophones (ETA) et (iv) des diplômes (DIP).
Les cours sont offerts à des étudiants ($\exists a \sqsubseteq C$ et $\exists a^- \sqsubseteq ET$). Les cours de langues sont offerts à des non anglophones ($la \sqsubseteq a$, $\exists la \sqsubseteq L$ et $\exists la^- \sqsubseteq \neg ETA$). Il n'y a pas de contraintes sur les étudiants pour les cours de maths ($ma \sqsubseteq a$ et $\exists ma \sqsubseteq M$).
Les mathématiciens enseignent des cours de maths ($\exists ens \sqsubseteq ENM$ et $\exists ens^- \sqsubseteq M$). Les étudiants sont enseignés par des enseignants et les étudiants et les enseignants sont disjoints ($\exists par \sqsubseteq ET$, $\exists par^- \sqsubseteq EN$ et $ET \sqsubseteq \neg EN$). Enfin, les étudiants valident des diplômes ($\exists val \sqsubseteq ET$, $\exists val^- \sqsubseteq DIP$) et les diplômes sont préparés par des étudiants ($\exists ppar \sqsubseteq DIP$ et $\exists ppar^- \sqsubseteq ET$). Un exemple d'une Abox de $\mathbf{K}$ pourrait être $\mathcal{A} = \{L(Ar), ETA(Marc), la(Fr, Marc)\}$ ce qui signifie que le centre offre un cours de langue arabe et que Marc est un étudiant inscrit au cours de langue française. Une requête sur $\mathbf{K}$ pourrait être $q(x) = C(x)$ pour s'informer sur les cours offerts par le centre ou $q(x) = \exists y\ L(x) \wedge a(x, y)$ permettant de savoir quels sont les cours de langues offerts et ayant des étudiants inscrits. ■

$L \sqsubseteq C$	$la \sqsubseteq a$	$\exists par \sqsubseteq ET$
$M \sqsubseteq C$	$\exists la \sqsubseteq L$	$\exists par^- \sqsubseteq EN$
$L \sqsubseteq \neg M$	$\exists la^- \sqsubseteq \neg ETA$	$ET \sqsubseteq \neg EN$
$ENM \sqsubseteq EN$	$ma \sqsubseteq a$	$\exists val \sqsubseteq ET$
$ETA \sqsubseteq ET$	$\exists ma \sqsubseteq M$	$\exists val^- \sqsubseteq DIP$
$\exists a \sqsubseteq C$	$\exists ens \sqsubseteq ENM$	$\exists ppar \sqsubseteq DIP$
$\exists a^- \sqsubseteq ET$	$\exists ens^- \sqsubseteq M$	$\exists ppar^- \sqsubseteq ET$

FIGURE 4.1 – La Tbox $\mathcal{T}$ de $\mathbf{K}$

4.2 Vérification de la consistance

Dans [25], la vérification de la consistance d'une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ s'effectue en calculant tout d'abord la clôture des NI de $\mathcal{T}$, notée $cln(\mathcal{T})$. Celle-ci est ensuite transformée en une union (booléenne) de requêtes conjonctives dont l'évaluation sur $\mathcal{A}$, considérée comme une base de données relationnelles $db(\mathcal{A})$, donnera la réponse à la vérifiation de consistance. Pour cela, la requête booléenne modélise la recherche de contre-exemples aux NI de $cln(\mathcal{T})$.

Le calcul de $cln(\mathcal{T})$ est effectuée de la manière suivante.

Définition 8 (Clôture des NI) *Soit $\mathcal{T}$ une TBox en* DL-LITE$_\mathcal{R}$*. La clôture des NI de $\mathcal{T}$, notée par $cln(\mathcal{T})$, est la TBox définie comme suit :*

1. *Toutes les NI de $\mathcal{T}$ sont aussi dans $cln(\mathcal{T})$.*
2. *Si $B_1 \sqsubseteq B_2$ est dans $\mathcal{T}$ et $B_2 \sqsubseteq \neg B_3$ ou $B_3 \sqsubseteq \neg B_2$ est dans $cln(\mathcal{T})$, alors $B_1 \sqsubseteq \neg B_3$ est dans $cln(\mathcal{T})$.*
3. *Si $R_1 \sqsubseteq R_2$ est dans $\mathcal{T}$ et $\exists R_2 \sqsubseteq \neg B$ ou $B \sqsubseteq \neg \exists R_2$ est dans $cln(\mathcal{T})$, alors $\exists R_1 \sqsubseteq \neg B$ est dans $cln(\mathcal{T})$.*
4. *Si $R_1 \sqsubseteq R_2$ est dans $\mathcal{T}$ et $\exists R_2^- \sqsubseteq \neg B$ ou $B \sqsubseteq \neg \exists R_2^-$ est dans $cln(\mathcal{T})$, alors $\exists R_1^- \sqsubseteq \neg B$ est dans $cln(\mathcal{T})$.*
5. *Si $R_1 \sqsubseteq R_2$ est dans $\mathcal{T}$ et $R_2 \sqsubseteq \neg R_3$ ou $R_3 \sqsubseteq \neg R_2$ est dans $cln(\mathcal{T})$, alors $R_1 \sqsubseteq \neg R_3$ est dans $cln(\mathcal{T})$.*
6. *Si $R_1 \sqsubseteq \neg R_2$ ou $R_2 \sqsubseteq \neg R_1$ sont dans $cln(\mathcal{T})$ alors $R_1^- \sqsubseteq \neg R_2^-$ est dans $cln(\mathcal{T})$.*
7. *Si une des assertions $\exists R \sqsubseteq \neg \exists R$, $\exists R^- \sqsubseteq \neg \exists R^-$, ou $R \sqsubseteq \neg R$ est dans $cln(\mathcal{T})$, alors ces trois assertions sont dans $cln(\mathcal{T})$.*

Exemple. 11 (Clôture des NI d'une Tbox)
La NI $\exists ens \sqsubseteq \neg \exists a^-$ est une NI logiquement impliquée par $\mathcal{T}$. Montrons comment elle est ajoutée à $cln(\mathcal{T})$ par les règles de la Définition 8. Notons que cette NI modélise le fait que le domaine du rôle *ens* (des enseignants mathématiciens) est disjoint du co-domaine du rôle a (des étudiants).

Par la règle 1, $cln(\mathcal{T})$ contient initialement toutes les NI de $\mathcal{T}$: $cln(\mathcal{T}) = \{L \sqsubseteq \neg M, \exists la^- \sqsubseteq \neg ETA, ET \sqsubseteq \neg EN\}$. La règle 2 s'applique à $\exists a^- \sqsubseteq ET \in \mathcal{T}$ et $ET \sqsubseteq \neg EN \in cln(\mathcal{T})$ et ajoute $\exists a^- \sqsubseteq \neg EN$ à $cln(\mathcal{T})$. La règle 2 s'applique alors de nouveau à $ENM \sqsubseteq EN$ et $\exists a^- \sqsubseteq \neg EN$ ce qui ajoute $ENM \sqsubseteq \neg \exists a^-$ à $cln(\mathcal{T})$. Enfin, la règle 2 s'applique à $\exists ens \sqsubseteq ENM$ et $ENM \sqsubseteq \neg \exists a^-$, et $\exists ens \sqsubseteq \neg \exists a^-$ est ajouté à $cln(\mathcal{T})$. ■

La fonction de traduction δ d'une NI en une requête conjonctive booléenne recherchant ses (éventuels) contre-exemples est définie par :

- $\delta(B_1 \sqsubseteq \neg B_2) = \exists x\ (\gamma_1(x) \wedge \gamma_2(x))$ tel que
  - $\gamma_i(x) = A_i(x)$ si $B_i = A_i$,
  - $\gamma_i(x) = \exists y_i\ P_i(x, y_i)$ si $B_i = \exists P_i$ et
  - $\gamma_i(x) = \exists y_i\ P_i(y_i, x)$ si $B_i = \exists P_i^-$;

- $\delta(R_1 \sqsubseteq \neg R_2) = \exists x, y\ (\rho_1(x, y) \wedge \rho_2(x, y))$ tel que
  - $\rho_i(x, y) = P_i(x, y)$ si $R_i = P_i$ et
  - $\rho_i(x, y) = P_i(y, x)$ si $R_i = P_i^-$.

Exemple. 12 (Traduction d'une NI)
Considérons la NI $\exists ens \sqsubseteq \neg\exists a^-$ de $cln(\mathcal{T})$ (*cf.* Exemple 11).
Sa traduction en une requête conjonctive booléenne recherchant ses (éventuels) contre-exemples est : $\delta(\exists ens \sqsubseteq \exists a^-) = \exists x(\exists y_1\ ens(x, y_1) \wedge \exists y_2\ a(y_2, x))$. ■

Pour mettre en œuvre le principe de vérification de la consistance décrit ci-dessus, [25] propose l'algorithme *Consistent* (Algorithme 3). Celui-ci prend en entrée une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ et construit une requête booléenne q_{unsat} servant à vérifier si les concepts et rôles devant être disjoints, d'après les connaissances intentionnelles modélisées dans $\mathcal{T}$, ont bien des instances disjointes dans $\mathcal{A}$. Pour cela, cette requête tente d'exhiber s'il existe dans $\mathcal{A}$ des contre-exemples aux NI impliquées logiquement par $\mathcal{T}$ (lignes 4 à 6). q_{unsat} est obtenue en faisant l'union des *traductions* (lignes 1 à 3) des NI de $cln(\mathcal{T})$.

Algorithme 3: L'algorithme *Consistent* original
Consistent(**K**)
Entrée: une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Sortie: *vrai* si **K** est satisfiable, $faux$ sinon
(1) $q_{unsat} := \{\bot\}$ (c-à-d. q_{unsat} est initialisée à $faux$)
(2) **foreach** $\alpha \in cln(\mathcal{T})$
(3) $\quad q_{unsat} := q_{unsat} \cup \{\delta(\alpha)\}$
(4) **if** $q_{unsat}^{db(\mathcal{A})} = \emptyset$
(5) $\quad$ **return** *vrai*
(6) **else return** $faux$

Exemple. 13 (Vérification de la consistance)
Considérons la BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ de la Figure 4.1, où $\mathcal{A} = \{a(algebre, Marc), par(Pierre, Vincent), ens(Marc, Analyse), ppar(CS, Pierre)\}$.

Consistent(**K**) calcule dans un premier temps $cln(\mathcal{T})$. Celle-ci contient la NI $\exists ens \sqsubseteq \neg\exists a^-$ – à cause de $\exists a^- \sqsubseteq ET$, $ET \sqsubseteq \neg EN$, $ENM \sqsubseteq EN$, et $\exists ens \sqsubseteq ENM$ – dont la traduction en logique de premier ordre est $\exists x(\exists y_1\ ens(x, y_1) \wedge \exists y_2\ a(y_2, x))$. Par conséquent, $q_{unsat}^{db(\mathcal{A})}$ s'évalue à *vrai* et *Consistent*(**K**) retourne $faux$. En effet, le problème vient du fait que les étudiants et les enseignants doivent être disjoints selon la Tbox ($ET \sqsubseteq \neg EN$), ce qui induit la NI $\exists ens \sqsubseteq \neg\exists a^-$ de $cln(\mathcal{T})$, alors qu'il existe un étudiant "Marc" auquel le centre offre un cours d'algèbre et qui est en même temps enseignant d'un cours d'analyse.

Si nous considérons maintenant la Abox $\mathcal{A} = \{a(Algebre, Marc), par(Pierre, Vincent), ens(Vincent, Analyse)\}$, $Consistent(\mathbf{K})$ retourne *vrai* car aucune NI de $cln(\mathcal{T})$ n'est violée par $\mathcal{A}$. ■

La Propriété 5 formalise des résultats importants – pour la suite de cette thèse – sur la vérification de la consistance en DL-LITE$_\mathcal{R}$ (*cf.* [25]).

Propriété 5 *Soit* $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *une BC* DL-LITE$_\mathcal{R}$ *:*
- $\mathbf{K}$ *est consistante si et seulement si* $db(\mathcal{A})$ *est un modèle de* $\langle cln(\mathcal{T}), \mathcal{A} \rangle$ *;*
- *si* α *est une NI et* $\mathcal{T} \models \alpha$*, alors* $cln(\mathcal{T}) \models \alpha$ *;*
- *si* α *est une NI et* $cln(\mathcal{T}) \models \alpha$*, alors* $\mathcal{T} \models \alpha$ *;*
- $Consistent(\mathbf{K})$ *termine, et* $\mathbf{K}$ *est consistante si et seulement si* $Consistent(\mathbf{K}) = true$*.*

4.3 Répondre à des requêtes

Dans [25], le calcul des réponses à une requête s'effectue par reformulation de la requête en fonction des connaissances de la Tbox, puis par l'évaluation de ses reformulations sur la Abox considérée comme une simple base de données relationnelles. Intuitivement, l'idée est de compiler les connaissances de la Tbox dans la requête, afin de se ramener à un problème classique d'évaluation de requêtes en bases de données.

4.3.1 Reformulation de requêtes

La reformulation d'une requête conjonctive q sur une Tbox $\mathcal{T}$ s'effectue en utilisant les PI de $\mathcal{T}$ comme règles de réécritures. Les PI sont vues comme des règles logiques pouvant être appliquées en chaînage arrière aux atomes de la requête. Plus précisément, une PI I est *applicable à un atome* $A(x)$ d'une requête si I a A dans sa partie droite, et, une PI I est *applicable à un atome* $P(x_1, x_2)$ d'une requête si (i) $x_2 = \_$ et la partie droite de I est $\exists P$; ou (ii) $x_1 = \_$ et la partie droite de I est $\exists P^-$; ou (iii) I est une inclusion de rôles et sa partie droite est P ou P^-. Notons que _ symbolise ici une variable existentielle n'apparaissant qu'une seule fois dans la requête (c-à-d. il n'y a pas de jointure sur cette variable).

La Définition suivante définit le résultat $gr(g, I)$ de l'application de la PI I à l'atome g. Cette opération est centrale pour la reformulation d'une requête.

Définition 9 (Application d'une PI à un atome) *Soit* I *une inclusion applicable à un atome* g*.* $gr(g, I)$ *est l'atome définit comme suit :*

1. *si* $g = A(x)$ *et* $I = A_1 \sqsubseteq A$ *alors* $gr(g, I) = A_1(x)$
2. *si* $g = A(x)$ *et* $I = \exists P \sqsubseteq A$ *alors* $gr(g, I) = P(x, \_)$
3. *si* $g = A(x)$ *et* $I = \exists P^- \sqsubseteq A$ *alors* $gr(g, I) = P(\_, x)$
4. *si* $g = P(x, \_)$ *et* $I = A \sqsubseteq \exists P$ *alors* $gr(g, I) = A(x)$
5. *si* $g = P(x, \_)$ *et* $I = \exists P_1 \sqsubseteq \exists P$ *alors* $gr(g, I) = P_1(x, \_)$
6. *si* $g = P(x, \_)$ *et* $I = \exists P_1^- \sqsubseteq \exists P$ *alors* $gr(g, I) = P_1(\_, x)$
7. *si* $g = P(\_, x)$ *et* $I = A \sqsubseteq \exists P^-$ *alors* $gr(g, I) = A(x)$
8. *si* $g = P(\_, x)$ *et* $I = \exists P_1 \sqsubseteq \exists P^-$ *alors* $gr(g, I) = P_1(x, \_)$
9. *si* $g = P(\_, x)$ *et* $I = \exists P_1^- \sqsubseteq \exists P^-$ *alors* $gr(g, I) = P_1(\_, x)$

10. si $g = P(x_1, x_2)$ *et soit* $I = P_1 \sqsubseteq P$ *ou* $I = P_1^- \sqsubseteq P^-$ *alors* $gr(g, I) = P_1(x_1, x_2)$
11. si $g = P(x_1, x_2)$ *et soit* $I = P_1 \sqsubseteq P^-$ *ou* $I = P_1^- \sqsubseteq P$ *alors* $gr(g, I) = P_1(x_2, x_1)$

$PerfectRef$ (Algorithme 4) est l'algorithme proposé par [25] pour le calcul des reformulations. Étant donné une requête conjonctive q, $PerfectRef$ la reformule en utilisant les PI de $\mathcal{T}$ suivant la Définition 9 (boucle (a), lignes 5 à 7).

Le point subtil de $PerfectRef$ est le besoin de simplifier les reformulations produites (boucle (b), lignes 8 à 10), de sorte que des PI qui n'étaient pas applicables à une reformulation deviennent éventuellement applicables à certaines de ses simplifications. Une simplification consiste à unifier deux atomes d'une reformulation en utilisant leur *unificateur le plus général* (à l'aide de *reduce*, ligne 10), puis à remplacer les variables existentielles n'apparaissant plus qu'une seule fois par _ (à l'aide de τ, ligne 10).

Algorithme 4: L'algorithme $PerfectRef$ original
$PerfectRef(q, \mathcal{T})$
Entrée: une requête conjonctive q et une Tbox $\mathcal{T}$
Sortie: une union de requêtes conjonctives
(1) $PR := \{q\}$
(2) **repeat**
(3) $PR' := PR$
(4) **foreach** $q \in PR'$
(5) (a) **foreach** $g \in q$
(6) **if** I est applicable à g
(7) $PR := PR \cup \{q[g/gr(g, I)]\}$
(8) (b) **foreach** $g_1, g_2 \in q$
(9) **if** g_1 et g_2 sont unifiables
(10) $PR := PR \cup \{\tau(reduce(q, g_1, g_2))\}$
(11) **until** $PR' = PR$

La Propriété 6 formalise des résultats importants – pour la suite de cette thèse – sur la reformulation des requêtes en DL-LITE$_\mathcal{R}$ (*cf.* [25]).

Propriété 6 *Soient* $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *une BC* DL-LITE$_\mathcal{R}$ *et* q *une requête conjonctive sur* $\mathcal{T}$ *:*
- $PerfectRef(q, \mathcal{T})$ *termine,*
- $PerfectRef(q, \mathcal{T})$ *s'exécute en temps polynomial dans la taille de* $\mathcal{T}$ *et*
- $PerfectRef(q, \mathcal{T}) \cup T \models q$.

4.3.2 Évaluation des reformulations

Pour calculer l'ensemble $ans(Q, \mathbf{K})$ des réponses à une union de requêtes conjonctives Q sur une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, [25] propose l'algorithme *Answer* (Algorithme 5). Il vérifie tout d'abord si $\mathbf{K}$ est inconsistante (ligne 1) et retourne dans ce cas l'ensemble fini $AllTup(Q, \mathbf{K})$ de tous les tuples d'arité de Q pouvant être générés à partir des constantes de $\mathcal{A}$ (ligne 2). Si $\mathbf{K}$ est consistance, $ans(Q, \mathbf{K})$ est calculé en évaluant sur $\mathcal{A}$ – considérée comme une base de données relationnelles $db(\mathcal{A})$– l'union de requêtes conjonctives obtenue par reformulation de Q (ligne 3).

Algorithme 5: L'algorithme *Answer* original
$Answer(Q, \mathbf{K})$
Entrée: une union de requêtes conjonctives Q et une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Sortie: $ans(Q, \mathbf{K})$
(1) **if** not $Consistent(\mathbf{K})$
(2) **return** $Alltup(Q, \mathbf{K})$
(3) **else return** $(\bigcup_{q_i \in Q} PerfectRef(q_i, \mathcal{T}))^{db(\mathcal{A})}$

La Propriété 7 formalise des résultats importants – pour la suite de cette thèse – sur l'évaluation des reformulations de requêtes en DL-LITE$_\mathcal{R}$ (*cf.* [25]).

Propriété 7 *Soient* $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *une BC* DL-LITE$_\mathcal{R}$ *et* Q *une union de requêtes conjonctives.*
- $Answer(Q, \mathbf{K})$ *termine ;*
- $ans(Q, \mathbf{K}) = Answer(Q, \mathbf{K})$ *;*
- *Si* PR *est l'union de requêtes conjonctives retournée par* $PerfectRef(q, \mathcal{T})$ *où* $q \in Q$ *alors pour toute Abox* DL-LITE$_\mathcal{R}$ $\mathcal{A}$ *telle que* $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *est satisfiable,* $ans(q, \mathbf{K}) = PR^{db(\mathcal{A})}$.
- *Répondre à des unions de requêtes conjonctives dans* DL-LITE$_\mathcal{R}$ *est PTime dans la taille de la Tbox, et LogSpace dans la taille de la Abox.*

Exemple. 14 (Réponse aux requêtes en DL-LITE$_\mathcal{R}$**)**
Considérons la BC de la Figure 4.1 $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ avec $\mathcal{A} = \{a(Algebre, Marc), par(Pierre, Vincent), ens(Marc, Analyse), ppar(CS, Pierre)\}$.
Étant donnée la requête $q(x) = DIP(x)$, $Answer(q, \mathbf{K})$ teste tout d'abord si $\mathbf{K}$ est consistante. $Consistent(\mathbf{K})$ retourne $faux$ (*cf.* l'Exemple 13). $Answer(q, \mathbf{K})$ retourne donc $Alltup(q, \mathbf{K})$.
Si nous considérons maintenant la Abox $\mathcal{A} = \{a(Algebre, Marc), par(Pierre, Vincent), ens(Vincent, Analyse)\}$ alors $\mathbf{K}$ est consistante (*cf.* l'Exemple 13) et $Answer(q, \mathbf{K})$ calcule la reformulation $\{DIP(x)\} \cup \{ppar(x, \_)\} \cup \{val(\_, x)\}$ puisque $\exists ppar \sqsubseteq DIP$ et $\exists val^- \sqsubseteq DIP$ sont applicables à $DIP(x)$. $Answer(q, \mathbf{K})$ retourne donc $\{(CS)\}$. ■

4.4 Vérification de la consistance par reformulation

Un autre moyen de tester la consistance d'une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ est de procéder par reformulation en utilisant l'algorithme $PerfectRef$ de la Section 4.3.1. En effet, il est montré dans [6] qu'on peut considérer seulement les NI de $\mathcal{T}$ (c-à-d. on ne calcule pas $cln(\mathcal{T})$) et les transformer par la fonction de traduction δ en une union booléenne de requêtes conjonctives Q modélisant la recherche de leurs éventuels contre-exemples dans $\mathcal{A}$. La reformulation par $PerfectRef$ des requêtes conjonctives contenues dans Q modélise alors toutes les façons de trouver d'éventuels contre-exemples aux NI de $\mathcal{T}$ dans $\mathcal{A}$. L'évaluation de cette requête sur $\mathcal{A}$, vue comme une base de données relationnelles, permet alors de vérifier la consistance de $\mathbf{K}$. L'algorithme 6 illustre cette alternative au test de consistance vu précédemment.

Exemple. 15 (Vérification de la consistance via $PerfectRef$**)**
Nous avons vu dans la Section 4.2 (*cf.* Exemple 13) que si $\mathcal{A} = \{a(algebre, Marc), par(Pierre, Vincent), ens(Marc, Analyse), ppar(CS, Pierre)\}$ de $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathbf{K}$ est inconsistante et que cette inconsistance a été détectée en évaluant la requête booléenne $\exists x(\exists y_1\, ens(x, y_1) \wedge \exists y_2\, a(y_2, x))$ modélisant la recherche de contre-exemples à la NI $\exists ens \sqsubseteq \neg \exists a^-$ de $cln(\mathcal{T})$.

Montrons que cette inconsistance peut être détectée par reformulation. $\mathcal{T}$ contient les NI suivantes :

Algorithme 6: L'algorithme *Consistent* via reformulation
Consistent(**K**)
Entrée: une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Sortie: *vrai* si **K** est satisfiable, $faux$ sinon
(1) $q_{unsat} := \{\bot\}$ (c-à-d. q_{unsat} est initialisée à $faux$)
(2) **foreach** $\alpha \in \mathcal{T}$
(3) $\quad q_{unsat} := q_{unsat} \cup PerfectRef(\delta(\alpha), \mathcal{T})$
(4) **if** $q_{unsat}^{db(\mathcal{A})} = \emptyset$
(5) $\quad$ **return** *vrai*
(6) **else return** $faux$

- $L \sqsubseteq \neg M$ dont la traduction est $q_1() = \exists x L(x) \wedge M(x)$,
- $\exists la^- \sqsubseteq \neg ETA$ dont la traduction est $q_2() = \exists x \exists y la(y, x) \wedge ETA(x)$ et
- $ET \sqsubseteq \neg EN$ dont la traduction est $q_3() = \exists x ET(x) \wedge EN(x)$.

Considérons la requête Q correspondant à l'union de q_1, q_2 et q_3. L'évaluation des *reformulations* de Q mène – entre autres – à l'évaluation de $PerfectRef(q_3, \mathcal{T})$, c-à-d. – entre autres – de la reformulation de q_3 : $q_3' = a(\_, x) \wedge ens(x, \_)$. L'évaluation de q_3' est positivie (pour $x = Marc$), ce qui exhibe un contre-exemple à la NI $ET \sqsubseteq \neg EN$ de $\mathcal{T}$, et donc que **K** est inconsistante. ■

Chapitre 5

Gestion décentralisée de données en DL-LITE$_\mathcal{R}$

Pour des raisons de passage à l'échelle, de robustesse et de protection de données, il et important d'étudier un modèle de données totalement décentralisé pour le Web Sémantique vu comme un immense système pair-à-pair de gestion de données (PDMS). Chaque pair peut avoir sa propre ontologie pour décrire ses données et interagit avec d'autres pairs en établissant des connexions sémantiques – *mappings* – avec leurs ontologies. Le résultat est un réseau de pairs avec aucune connaissance centralisée et donc aucun contrôle global des données et connaissances distribuées sur le Web.

Nous présentons dans ce chapitre la troisième contribution de cette thèse : un modèle distribué de données en DL-LITE$_\mathcal{R}$ et les algorithmes décentralisés associés de gestion de données. Nous revisitons l'approche centralisée de [25] présentée dans le Chapitre 4, pour la consistance des données et la réponse à des requêtes par reformulation, dans le but de définir les algorithmes décentralisés correspondants. Notre approche consiste à réduire ces problèmes en des raisonnements décentralisés en logique propositionnelle (LP) et utiliser l'algorithme DECA$_{LR}$ présenté dans le Chapitre 2 pour raisonner au niveau propositionnel. Ceci nous permet de déployer des PDMS DL-LITE$_\mathcal{R}$ au dessus des P2PIS propositionnels présentés dans le Chapitre 2.

Le chapitre est organisé comme suit. Nous définissons les PDMS DL-LITE$_\mathcal{R}$ auxquels nous nous intéressons dans la Section 5.1. Nous présentons ensuite l'encodage d'un PDMS DL-LITE$_\mathcal{R}$ en logique propositionnelle dans la Section 5.2. Nous abordons le problème de vérification décentralisée de la consistance dans la Section 5.3 et le problème de répondre à des requêtes d'une manière décentralisée dans la Section 5.4. Enfin, nous présentons les principaux travaux de la littérature sur les PDMS dans la Section 5.5.

5.1 Les systèmes de gestion de données pair-à-pair en DL-LITE$_\mathcal{R}$

Un PDMS $\mathcal{S}$ DL-LITE$_\mathcal{R}$ est un ensemble de pairs $\{\mathcal{P}_i\}_{i=1..n}$, où l'indice i modélise l'identifiant du pair $\mathcal{P}_i$ (par exemple son adresse IP). Chaque pair $\mathcal{P}_i$ gère sa propre BC DL-LITE$_\mathcal{R}$ $\mathbf{K}_i = \langle \mathcal{T}_i, \mathcal{A}_i \rangle$ exprimée en termes de son propre *vocabulaire*, c-à-d. ses concepts et rôles atomiques. Nous notons A_i (resp. P_i) le concept atomique A (resp. le rôle atomique P) de $\mathcal{P}_i$.

Des pairs peuvent être connectés deux-à-deux par des *mappings* qui sont des inclusions (PI et/ou NI) utilisant un concept ou un rôle de chacun des pairs. Pour simplifier, nous considérons

qu'un mapping est stocké dans la Tbox des deux pairs mis en jeu.

D'un point de vue logique, un PDMS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ est une BC DL-LITE$_\mathcal{R}$ standard (bien que distribuée) $\mathbf{K} = \bigcup_{i=1}^{n} \mathbf{K}_i$. C'est-à-dire que, contrairement à d'autres approches (par exemple [26]), nous adoptons une sémantique logique standard pour les mappings.

Exemple. 16 (Service décentralisé de cours particuliers)
La Figure 5.1 représente le PDMS $\mathcal{S}$ obtenu par décentralisation de la BC centralisée de l'Exemple 10 du Chapitre 4 sur les quatre pairs $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ et $\mathcal{P}_4$ où $\mathcal{P}_1$ gère les cours, $\mathcal{P}_2$ les étudiants, $\mathcal{P}_3$ les enseignants et $\mathcal{P}_4$ les diplômes. Les BC des pairs sont les nœuds étiquetés par les noms des pairs et les mappings entre les pairs étiquettent l'arête qui lie leur BC respective.

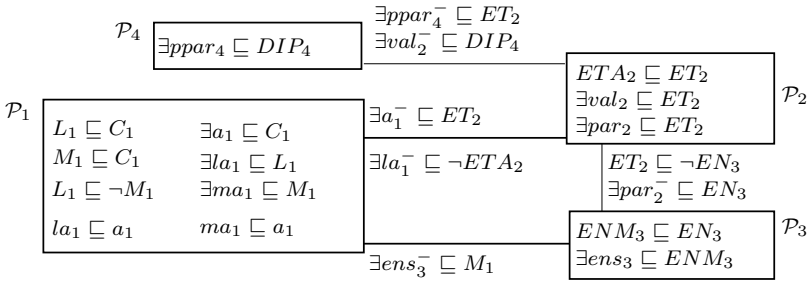

FIGURE 5.1 – Le PDMS $\mathcal{S}$

■

5.2 Encodage propositionnel d'une Tbox

Notre approche s'appuie sur la réduction des problèmes de gestion de données à des raisonnements en LP. Pour cela, nous encodons un PDMS DL-LITE$_\mathcal{R}$ (plus précisément sa Tbox) en une théorie décentralisée de LP.

L'encodage d'une Tbox, noté $\Phi(\mathcal{T})$, est la théorie de LP correspondant à l'union des encodages des inclusions de $\mathcal{T}$: $\Phi(\mathcal{T}) = \bigcup_{I \in \mathcal{T}} \Phi(I)$.

L'encodage d'une inclusion de concepts $B \sqsubseteq C$, noté $\Phi(B \sqsubseteq C)$ est défini récursivement par $\{\Phi(B) \Rightarrow \Phi(C)\}$ où l'encodage des concepts B et C est $\Phi(B) = A$ lorsque $B = A$, $\Phi(B) = P^{\exists}$ lorsque $B = \exists P$, $\Phi(B) = P^{\exists^-}$ lorsque $B = \exists P^-$, $\Phi(C) = \Phi(B)$ lorsque $C = B$ et $\Phi(C) = \neg\Phi(B)$ lorsque $C = \neg B$.

L'encodage d'une inclusion de rôles $R \sqsubseteq E$, noté $\Phi(R \sqsubseteq E)$, est défini comme suit :
- $\Phi(P \sqsubseteq Q)$=$\{P \Rightarrow Q, P^- \Rightarrow Q^-, P^{\exists} \Rightarrow Q^{\exists}, P^{\exists^-} \Rightarrow Q^{\exists^-}\}$
- $\Phi(P^- \sqsubseteq Q)$=$\{P^- \Rightarrow Q, P \Rightarrow Q^-, P^{\exists^-} \Rightarrow Q^{\exists}, P^{\exists} \Rightarrow Q^{\exists^-}\}$
- $\Phi(P \sqsubseteq Q^-)$=$\{P \Rightarrow Q^-, P^- \Rightarrow Q, P^{\exists} \Rightarrow Q^{\exists^-}, P^{\exists^-} \Rightarrow Q^{\exists}\}$
- $\Phi(P^- \sqsubseteq Q^-)$=$\{P^- \Rightarrow Q^-, P \Rightarrow Q, P^{\exists} \Rightarrow Q^{\exists}, P^{\exists^-} \Rightarrow Q^{\exists^-}\}$
- $\Phi(P \sqsubseteq \neg Q)$=$\{P \Rightarrow \neg Q, P^- \Rightarrow \neg Q^-\}$
- $\Phi(P^- \sqsubseteq \neg Q)$=$\{P^- \Rightarrow \neg Q, P \Rightarrow \neg Q^-\}$
- $\Phi(P \sqsubseteq \neg Q^-)$=$\{P \Rightarrow \neg Q^-, P^- \Rightarrow \neg Q\}$
- $\Phi(P^- \sqsubseteq \neg Q^-)$=$\{P^- \Rightarrow \neg Q^-, P \Rightarrow \neg Q\}$

Enfin, l'encodage d'une Tbox distribuée $\bigcup_{i=1}^{n} \mathcal{T}_i$ d'un PDMS est la théorie propositionnelle distribuée $\bigcup_{i=1}^{n} \Phi(\mathcal{T}_i)$ d'un P2PIS propositionnel (*cf.* Chapitre 2) obtenue en encodant chaque Tbox locale $\mathcal{T}_i$.

Dans la suite de ce chapitre, nous aurons besoin d'encoder des rôles de la façon suivante. L'encodage d'un rôle E est $\Phi(E) = \Phi(R)$ lorsque $E = R$, $\Phi(E) = \neg\Phi(R)$ lorsque $E = \neg R$, $\Phi(R) = P$ lorsque $R = P$ et $\Phi(R) = P^-$ lorsque $R = P^-$.

5.3 Vérification décentralisée de la consistance

Notre approche consiste à décentraliser le calcul de la clôture des NI d'une Tbox présenté dans le Chapitre 4. Plus précisément, nous proposons un calcul décentralisé de la clôture des NI de la Tbox distribuée d'un PDMS *sans rôle vide* (c-à-d. où aucune contrainte n'interdit d'avoir des instances pour des rôles : $R \sqsubseteq \neg R$, $\exists R \sqsubseteq \neg\exists R$ ou encore $\exists R^- \sqsubseteq \neg\exists R^-$). Nous reviendrons sur la limitation aux PDMS sans rôle vide en fin de section, et nous la lèverons ultérieurement grâce aux résultats de la prochaine section.

Le problème de vérification de la consistance peut se réduire en un raisonnement décentralisé en logique propositionnelle effectué sur l'encodage propositionnel de la Tbox. Nous commençons par établir le lien entre la déduction de NI dans une Tbox distribuée d'un PDMS et la déduction logique dans l'encodage propositionnel de cette Tbox. Nous précisons ensuite comment cette déduction peut être effectivement réalisée dans un cadre complètement décentralisé.

Theorème 5 (Clôture des NI et calcul de conséquences) *Soient $\mathcal{T}$ une Tbox distribuée d'un PDMS sans rôle vide et $\Phi(\mathcal{T})$ son encodage propositionnel. Soient X et Y deux concepts basiques ou deux rôles basiques distincts : $cln(\mathcal{T}) \models X \sqsubseteq \neg Y$ ssi $\Phi(\mathcal{T}) \models \neg\Phi(X) \vee \neg\Phi(Y)$.*

Preuve. ($\Rightarrow$) La preuve s'effectue par récurrence sur la taille minimale n de la séquence d'application de règles définissant la clôture des NI (Définition 8 du Chapitre 4) utilisées pour produire $X \sqsubseteq \neg Y$, la taille est minimale dans le sens où retirer une application de règle de la séquence ne permet plus de produire $X \sqsubseteq \neg Y$: si $cln(\mathcal{T})$ produit $X \sqsubseteq \neg Y$ après n applications de règles, $\Phi(\mathcal{T}) \models \neg\Phi(X) \vee \neg\Phi(Y)$.

Si $n = 0$, $X \sqsubseteq \neg Y$ est dans $\mathcal{T}$. Par construction, $\neg\Phi(X) \vee \neg\Phi(Y)$ est dans $\Phi(\mathcal{T})$, d'où $\Phi(\mathcal{T}) \models \neg\Phi(X) \vee \neg\Phi(Y)$.

Supposons que l'hypothèse de récurrence est vraie jusqu'à $n < r$ et montrons qu'elle est vraie pour $n = r$. Si $X \sqsubseteq \neg Y$ est ajouté à $cln(\mathcal{T})$ à $n = r$, alors c'est qu'une règle de clôture (*cf.* Definition 8), à l'exception de la règle 1 d'initialisation et de la règle 7 gérant les rôles vides, a pu être déclenchée à partir de $n = r - 1$. Ainsi, à $n = r - 1$, soit

1. $B_1 \sqsubseteq B_2$ est dans $\mathcal{T}$ et $B_2 \sqsubseteq \neg B_3$ (ou $B_3 \sqsubseteq \neg B_2$) est dans $cln(\mathcal{T})$, d'où $X \sqsubseteq \neg Y \equiv B_1 \sqsubseteq \neg B_3$. Dans ce cas, à $n = r - 1$, $\Phi(B_1 \sqsubseteq B_2)$ est dans $\Phi(\mathcal{T})$ (par construction) et $\Phi(\mathcal{T}) \models \Phi(B_2 \sqsubseteq \neg B_3)$ (ou $\Phi(\mathcal{T}) \models \Phi(B_3 \sqsubseteq \neg B_2)$) par hypothèse de récurrence. Par conséquent, $\Phi(\mathcal{T}) \models \neg B_1 \vee B_2$ et $\Phi(\mathcal{T}) \models \neg B_2 \vee \neg B_3$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \models \neg B_1 \vee \neg B_3$.
2. $R_1 \sqsubseteq R_2$ est dans $\mathcal{T}$ et $\exists R_2 \sqsubseteq \neg B$ (ou $B \sqsubseteq \neg\exists R_2$) est dans $cln(\mathcal{T})$, d'où $X \sqsubseteq \neg Y \equiv \exists R_1 \sqsubseteq \neg B$. Dans ce cas, à $n = r - 1$, $\Phi(R_1 \sqsubseteq R_2)$ est dans $\Phi(\mathcal{T})$ (par construction) et $\Phi(\mathcal{T}) \models \Phi(\exists R_2 \sqsubseteq \neg B)$ (ou $\Phi(\mathcal{T}) \models B \sqsubseteq \neg\exists R_2$) par hypothèse de récurrence. Par conséquent, $\Phi(\mathcal{T}) \models \neg R_1^{\exists} \vee R_2^{\exists}$ et $\Phi(\mathcal{T}) \models \neg R_2^{\exists} \vee \neg B$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \models \neg R_1^{\exists} \vee \neg B$.

3. $R_1 \sqsubseteq R_2$ est dans $\mathcal{T}$ et $\exists R_2^- \sqsubseteq \neg B$ (ou $B \sqsubseteq \neg\exists R_2^-$) est dans $cln(\mathcal{T})$, d'où $X \sqsubseteq \neg Y \equiv \exists R_1^- \sqsubseteq \neg B$. Dans ce cas, à $n = r - 1$, $\Phi(R_1 \sqsubseteq R_2)$ est dans $\Phi(\mathcal{T})$ (par construction) et $\Phi(\mathcal{T}) \models \Phi(\exists R_2^- \sqsubseteq \neg B)$ (ou $\Phi(\mathcal{T}) \models B \sqsubseteq \neg\exists R_2^-$) par hypothèse de récurrence. Par conséquent, $\Phi(\mathcal{T}) \models \neg R_1^{\exists -} \vee R_2^{\exists -}$ et $\Phi(\mathcal{T}) \models \neg R_2^{\exists -} \vee \neg B$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \models \neg R_1^{\exists -} \vee \neg B$.
4. $R_1 \sqsubseteq R_2$ est dans $\mathcal{T}$ et $R_2 \sqsubseteq \neg R_3$ (ou $R_3 \sqsubseteq \neg R_2$) est dans $cln(\mathcal{T})$, d'où $X \sqsubseteq \neg Y \equiv R_1 \sqsubseteq \neg R_3$. Dans ce cas, à $n = r - 1$, $\Phi(R_1 \sqsubseteq R_2)$ est dans $\Phi(\mathcal{T})$ (par construction) et $\Phi(\mathcal{T}) \models \Phi(R_2 \sqsubseteq \neg R_3)$ (ou $\Phi(\mathcal{T}) \models \Phi(R_3 \sqsubseteq \neg R_2)$) par hypothèse de récurrence. Par conséquent, $\Phi(\mathcal{T}) \models \neg R_1 \vee R_2$ et $\Phi(\mathcal{T}) \models \neg R_2 \vee \neg R_3$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \models \neg R_1 \vee \neg R_3$.
5. $R_1 \sqsubseteq \neg R_2$ est dans $cln(\mathcal{T})$ ou $R_2 \sqsubseteq \neg R_1$ est dans $cln(\mathcal{T})$, d'où $X \sqsubseteq \neg Y \equiv R_1^- \sqsubseteq \neg R_2^-$. Dans ce cas, à $n = r-1$, $\Phi(\mathcal{T}) \models \Phi(R_1 \sqsubseteq \neg R_2)$ ou $\Phi(\mathcal{T}) \models \Phi(R_2 \sqsubseteq \neg R_1)$. Par conséquent, $\Phi(\mathcal{T}) \models \neg R_1^- \vee R_2^-$ (*cf.* règles d'encodage de la Définition 5.2).

($\Leftarrow$) La preuve s'effectue par récurrence sur la plus petite taille n de preuve par résolution ($\vdash_R$ est complète pour les impliqués premiers) nécessaire pour produire l'impliqué premier $\neg\Phi(X) \vee \neg\Phi(Y)$ (tout autre impliqué découle d'un premier), la plus petite taille est dans le sens où retirer une étape de la preuve ne produit plus $\neg\Phi(X) \vee \neg\Phi(Y)$: si $\Phi(\mathcal{T}) \vdash_R \neg\Phi(X) \vee \neg\Phi(Y)$ avec une preuve de taille n, $cln(\mathcal{T}) \models X \sqsubseteq \neg Y$.

Si $n = 0$, $\neg\Phi(X) \vee \neg\Phi(Y)$ est dans $\Phi(\mathcal{T})$ (par construction). Par conséquent, d'après notre encodage propositionnel, $X \sqsubseteq \neg Y \in cln(\mathcal{T})$, donc $cln(\mathcal{T}) \models X \sqsubseteq \neg Y$.

Supposons que l'hypothèse de récurrence est vraie jusqu'à $n < r$ et montrons qu'elle est vraie pour $n = r$. Notons tout d'abord que si $\Phi(\mathcal{T}) \vdash_R \neg\Phi(X) \vee \neg\Phi(Y)$ alors $\neg\Phi(X) \vee \neg\Phi(Y)$ est l'encodage d'une NI DL-LITE$_\mathcal{R}$ (*cf.* les résolutions possibles entre les clauses d'un encodage propositionnel).
Si $\Phi(\mathcal{T}) \vdash_R \neg\Phi(X) \vee \neg\Phi(Y)$ est obtenue à $n = r$ résolutions, toutes les clauses purement négatives dérivées à $n = r - 1$ sont des NI logiquement impliquées par $cln(\mathcal{T})$ (hypothèse de récurrence), donc par $\mathcal{T}$ (*cf.* Proposition 5 du Chapitre 4). Puisque $\mathcal{T}$ implique logiquement la PI dont l'encodage a servi à dériver $\neg\Phi(X) \vee \neg\Phi(Y)$ à $n = r$, $\mathcal{T} \models X \sqsubseteq \neg Y$, d'où $cln(\mathcal{T}) \models X \sqsubseteq \neg Y$ (*cf.* Proposition 5). □

Le point subtil est que dans un cadre décentralisé, nous devons déclencher le calcul de la clôture des NI sur chaque pair – puisqu'aucun pair n'a une vision globale du PDMS – et ainsi initier le calcul à partir de PI et NI locales pouvant mener à la dérivation de nouvelles NI par interaction avec des NI et PI d'autres pairs. Pour cela, nous définissons (Définition 10) et calculons avec DECA$_{LR}$ (*cf.* Chapitre 2) la *clôture des NI d'un pair* par rapport à un PDMS sans rôle vide. Nous prouvons ensuite que la vérification de la consistance globale du PDMS revient à vérifier sa consistance par rapport à chacun des pairs.

Définition 10 (Clôture d'un pair par rapport à un PDMS) *Soit $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$ la Tbox distribuée d'un PDMS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ sans rôle vide. La* clôture des NI de $\mathcal{P}_i$ par rapport à $\mathcal{S}$, *notée $cln(\mathcal{P}_i)$, est obtenue de $\Phi(\mathcal{T})$ à l'aide de* DECA$_{LR}$ *comme suit :*

- *pour chaque PI $Z \sqsubseteq Y \in \mathcal{T}_i$ telle que Z est dans le vocabulaire de $\mathcal{P}_i$ et Y est dans celui de $\mathcal{P}_j$ (j peut être i), $Z \sqsubseteq \neg X \in cln(\mathcal{P}_i)$ pour tout $\neg\Phi(X) \in$ DECA$_{LR}{}^j(\Phi(Y))$.*
- *pour chaque NI $Z \sqsubseteq \neg Y \in \mathcal{T}_i$*
  - *si Z est dans le vocabulaire de $\mathcal{P}_i$ et Y dans celui de $\mathcal{P}_j$ (j peut être i), $Z \sqsubseteq \neg X \in cln(\mathcal{P}_i)$ pour chaque $\neg\Phi(X) \in$ DECA$_{LR}{}^j(\neg\Phi(Y))$*

– *si Y est dans le vocabulaire de $\mathcal{P}_i$ et Z dans celui de $\mathcal{P}_j$ (j peut être i), $X \sqsubseteq \neg Y \in cln(\mathcal{P}_i)$ pour tout $\neg\Phi(X) \in \text{DECA}_{LR}{}^j(\neg\Phi(Z))$.*

Pour vérifier la consistance du PDMS par rapport à un pair, nous fournissons une version décentralisée de l'algorithme *Consistent* original, notée $Consistent^i$ lorsqu'elle s'exécute sur le pair $\mathcal{P}_i$. Notre algorithme (Algorithme 7) est obtenu en remplaçant **foreach** $\alpha \in cln(\mathcal{T})$ à la ligne 2 de l'Algorithme 3 par **foreach** $\alpha \in cln(\mathcal{P}_i)$, et où q_{unsat} est évaluée par $\mathcal{P}_i$ sur le sous ensemble pertinent de toute la Abox – qui est inconnue – du PDMS. Par construction de q_{unsat}, chacune de ses requêtes conjonctives possède deux atomes, l'un issu de $\mathcal{P}_i$ et l'autre issu de $\mathcal{P}_j$ (j peut être i), le dernier fournissant dans le nom de son prédicat l'identifiant j du pair à contacter pour l'évaluation.

Algorithme 7: L'algorithme *Consistent* s'exécutant sur le pair $\mathcal{P}_i$ du PDMS $\mathcal{S}$
$Consistent^i()$
Entrée: aucune
Sortie: *vrai* si $\mathcal{S}$ est consistant par rapport à $\mathcal{P}_i$, $faux$ sinon
(1) $q_{unsat} := \bot$ (c-à-d. q_{unsat} est initialisée à $faux$)
(2) **foreach** $\alpha \in cln(\mathcal{P}_i)$
(3) $\quad q_{unsat} := q_{unsat} \vee \delta(\alpha)$
(4) **if** $q_{unsat}^{db(\bigcup_{i=1}^{k} \mathcal{A}_i)} = \emptyset$ tel que $1, \ldots, k$ sont tous les identifiants de pairs apparaissant dans q_{unsat}
(5) $\quad$ **return** *vrai*
(6) **else**
(7) $\quad$ **return** $faux$

Le Théorème 6 établit la correction d'exécuter localement $Consistent^i$ sur chaque pair $\mathcal{P}_i$ pour vérifier la consistance globale d'un PDMS sans rôle vide.

Theorème 6 (Correction de la vérification de la consistance) *Soit $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ un PDMS sans rôle vide. $\mathcal{S}$ est consistant ssi $Consistent^i$ retourne true pour tout $i = 1..n$.*

La preuve repose tout d'abord sur le Théorème 5 montrant l'équivalence entre la déduction d'une NI à partir d'une Tbox et la déduction de son encodage propositionnel à partir de l'encodage de la Tbox. Ensuite, la Propriété 5 du Chapitre 4 et la complétude de DECA_{LR} (prouvée dans le Chapitre 2) garantissent que $cln(\mathcal{P}_i)$ définie dans la Définition 10 contient toutes les NI impliquées par le PDMS faisant intervenir un concept ou rôle du vocabulaire du pair $\mathcal{P}_i$. Il est alors facile de voir qu'en exécutant $Consistent^i$ sur tout pair $\mathcal{P}_i$ du PDMS, nous obtenons toutes les NI impliquées par le PDMS. Enfin, la Propriété 5 du Chapitre 4 garantit que la vérification de consistance peut être faite en évaluant l'union de requêtes conjonctives q_{unsat} sur les parties pertinentes de la Abox. C'est exactement ce que fait d'une façon décentralisée l'exécution de $Consistent^i$ sur tous les pairs du PDMS.

PDMS DL-LITE$_\mathcal{R}$ avec rôles vides. Remarquons que nous n'avons considéré pour le moment que les PDMS DL-LITE$_\mathcal{R}$ sans rôles vides. C'est pourquoi dans la preuve du Théorème 5, nous n'avons pas traité la règle 7 de la Définition 8 du Chapitre 4 sur les rôles vides : si $\exists R \sqsubseteq \neg\exists R$, $\exists R^- \sqsubseteq \neg\exists R^-$ ou $R \sqsubseteq \neg R$ est dans $cln(\mathcal{T})$ alors ces trois assertions sont dans $cln(\mathcal{T})$.

En effet, notre première approche de la gestion de consistance ne permet pas de simuler cette règle. L'intuition est qu'elle n'encode pas une déduction fondée sur la résolution au premier ordre

entre une PI et une NI, mais simplement l'*ajout* de formules *en cours* de raisonnement pour la complétude de $cln(\mathcal{T})$. Voici un exemple illustrant la limite de notre approche en présence de rôles vides. Notons que cette limitation sera levée dans la section suivante.

La Figure 5.2 représente la Tbox d'un PDMS $\mathcal{S}$ formé par de deux pairs $\mathcal{P}_1$ et $\mathcal{P}_2$. Les BC des pairs sont les noeuds étiquetés par les noms des pairs et les mappings entre les pairs étiquettent l'arête qui lie leur BC respective. La Abox de $\mathcal{S}$ est $\mathcal{A} = \{A_1(a), B_1(a)\}$.

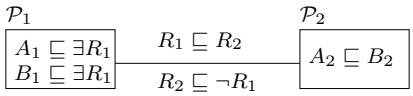

FIGURE 5.2 – Le PDMS avec rôles vides $\mathcal{S}$

Considérons ce PDMS comme une BC centralisée. $cln(\mathcal{T})$ contient initialement $R_2 \sqsubseteq \neg R_1$. En appliquant la règle 5 de la Définition 8 du Chapitre 4, $R_1 \sqsubseteq \neg R_1 \in cln(\mathcal{T})$, ce qui signifie que R_1 est un rôle vide. Ensuite, par la règle 7 de la Définition 8, $\exists R_1 \sqsubseteq \neg \exists R_1$ et $\exists R_1^- \sqsubseteq \neg \exists R_1^-$ s'ajoutent à $cln(\mathcal{T})$. $\exists R_1 \sqsubseteq \neg \exists R_1$ et $A_1 \sqsubseteq \exists R_1$ donnent $A_1 \sqsubseteq \neg \exists R_1$ par la règle 2 ce qui donne avec $B_1 \sqsubseteq \exists R_1$ l'inclusion $B_1 \sqsubseteq \neg A_1$. $\delta(B_1 \sqsubseteq \neg A_1) = \exists x A_1(x) \wedge B_1(x)$. En évaluant δ sur la Abox, on obtient la réponse "a" et la BC de $\mathcal{S}$ est donc inconsistante.

Considérons maintenant l'encodage propositionnel de ce PDMS et essayons de vérifier la consistance par calcul de conséquences. $\Phi(\mathcal{T}) = \{\neg A_1 \vee R_1^\exists, \neg B_1 \vee R_1^\exists, \neg R_1 \vee R_2, \neg R_1^- \vee R_2^-, \neg R_2^- \vee \neg R_1^-, \neg R_2 \vee \neg R_1, \neg A_2 \vee B_2\}$. Aucune déduction linéaire dans $\Phi(\mathcal{T})$ ne permet de produire $\neg A_1 \vee \neg B_1$ et donc l'inconsistance ne pourra pas être détectée. En effet, la raison est qu'ici nous n'avons pas accès à $\Phi(\exists R_1 \sqsubseteq \neg \exists R_1)$.

5.4 Répondre à des requêtes de façon décentralisée

Passons maintenant au problème de répondre à des requêtes dans les PDMS DL-LITE$_\mathcal{R}$. Nous avons vu dans le Chapitre 4 que pour calculer les réponses à une requête conjonctive q, l'algorithme *Answer* commence par tester la consistance de la BC, puis si la BC est consistante, il calcule les reformulations de la requête et les évalue sur la Abox considérée comme une simple base de données relationnelles.

Dans notre cadre distribué et complètement décentralisé, nous ne pouvons pas tester la consistance globale du PDMS : ceci impose de connaître tous les pairs du systèmes (*cf.* Théorème 6). En revanche, nous sommes en mesure de vérifier si un pair est dans un sous-PDMS consistant (*cf.* section 5.3).

Nous allons donc commencer par reformuler la requête puis nous vérifions la consistance du PDMS par rapport aux pairs apparaissant dans cette reformulation. Si le système est consistant par rapport à tous ces pairs, nous évaluons les reformulations et nous obtenons des réponses dites *bien fondées* puisqu'elles sont dérivées à partir d'un sous-ensemble consistant de la BC distribuée (éventuellement inconsistante) du PDMS.

5.4.1 Reformulation de requêtes

Comme pour la vérification décentralisée de la consistance, notre approche s'appuie sur l'encodage propositionnel et l'utilisation de DECA$_{LR}$ afin de décentraliser la *clôture en chaînage arrière par rapport à des PI* de chacun des atomes de la requête. La Définition 11 définit la *clô-*

ture en chaînage arrière d'un atome par rapport à des PI comme l'itération d'étapes élémentaires d'application de PI en chaînage arrière (*cf.* Définition 9 du Chapitre 4).

Définition 11 (Clôture en chaînage arrière d'un atome) *Soient PI un ensemble de PI, g un atome et $\mathcal{A}$ un ensemble d'atomes. La* clôture en chaînage arrière *de g par rapport à PI est $cl\_gr(g, PI) = \bigcup_{i\geq 1} cl\_gr^i(\{g\}, PI)$ où $cl\_gr^i(\{g\}, PI)$ est défini récursivement comme suit :*
- *$cl\_gr^1(\mathcal{A}, PI) = \{gr(g, I) \mid g \in \mathcal{A},\ I \in PI$ et I est applicable à $g\}$ et*
- *$cl\_gr^{i+1}(\mathcal{A}, PI) = cl\_gr^1(cl\_gr^i(\mathcal{A}, PI), PI)$.*

La Proposition 9 établit la terminaison de ce processus itératif. Sa preuve correspond à la preuve de terminaison de $PerfectRef$ (Lemme 34 dans [25]).

Proposition 9 (Terminaison de la clôture d'un atome) *La clôture en chaînage arrière d'un atome par rapport à un ensemble de PI est finie, c-à-d. il existe une constante n telle que $cl\_gr(g, PI) = \bigcup_{i=1}^{n} cl\_gr^i(\{g\}, PI)$.*

Le Théorème 7 est l'équivalent pour les PI de la propriété de transfert de l'encodage propositionnel pour les NI (Théorème 5).

Theorème 7 (Clôture en chaînage arrière via encodage) *Soit $\mathcal{T}$ une Tbox dont les PI forment l'ensemble PI. Soient g, g' des atomes et V_1, V_2 des variables propositionnelles.*
$g' \in cl\_gr(g, PI)$ ssi $\Phi(\mathcal{T}) \cup \{\neg V_1\} \models \neg V_2$ où :

1. *$g = A(x)$, $g' = A'(x)$, $V_1 = A$ et $V_2 = A'$;*
2. *$g = A(x)$, $g' = P(x, \_)$, $V_1 = A$ et $V_2 = P^{\exists}$;*
3. *$g = A(x)$, $g' = P(\_, x)$, $V_1 = A$ et $V_2 = P^{\exists^-}$;*
4. *$g = P(x, y)$, $g' = Q(x, y)$, $V_1 = P$ et $V_2 = Q$;*
5. *$g = P(x, y)$, $g' = Q(y, x)$, $V_1 = P$ et $V_2 = Q^-$;*
6. *$g = P(x, \_)$, $g' = A(x)$, $V_1 = P^{\exists}$ et $V_2 = A$;*
7. *$g = P(x, \_)$, $g' = Q(x, \_)$, $V_1 = P^{\exists}$ et $V_2 = Q^{\exists}$;*
8. *$g = P(x, \_)$, $g' = Q(\_, x)$, $V_1 = P^{\exists}$ et $V_2 = Q^{\exists^-}$;*
9. *$g = P(\_, x)$, $g' = A(x)$, $V_1 = P^{\exists^-}$ et $V_2 = A$;*
10. *$g = P(\_, x)$, $g' = Q(x, \_)$, $V_1 = P^{\exists^-}$ et $V_2 = Q^{\exists}$;*
11. *$g = P(\_, x)$, $g' = Q(\_, x)$, $V_1 = P^{\exists^-}$ et $V_2 = Q^{\exists^-}$.*

Preuve. ($\Rightarrow$) La preuve s'effectue par récurrence sur la taille minimale n de la séquence d'application de règles définissant la clôture des PI (Définition 9 du Chapitre 4) utilisées pour produire g', la taille est minimale dans le sens où retirer une application de règle de la séquence ne permet plus de produire g' : si $cl\_gr(g, PI)$ produit g' après n applications de règles, $\Phi(\mathcal{T}) \cup \{\neg V_1\} \models \neg V_2$.

Si $n = 0$, $g = g'$ d'où $V_1 \equiv \Phi(g)$ et $V_2 \equiv \Phi(g)$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg\Phi(g)\} \models \neg\Phi(g)$.

Supposons que l'hypothèse de récurrence est vraie jusqu'à $n < r$ et montrons qu'elle est vraie pour $n = r$. Si g' est ajoutée à $cl\_gr(g, PI)$ à $n = r$, alors c'est qu'une règle de réécriture (*cf.* Définition 9) a pu être déclenchée à partir de $n = r - 1$. Ainsi, à $n = r - 1$, soit

1. $A(x)$ est dans $cl\_gr(g, PI)$ et $A' \sqsubseteq A$ est dans PI, d'où $g' = A'(x)$ et $V_1 \equiv A$ et $V_2 \equiv A'$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(A)$ et $\Phi(\mathcal{T}) \models \Phi(A' \sqsubseteq A)$. Par conséquent, $\Phi(\mathcal{T}) \models A$ et $\Phi(\mathcal{T}) \models \neg A' \vee A$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg A\} \models \neg A'$.

2. $A(x)$ est dans $cl\_gr(g, PI)$ et $\exists P \sqsubseteq A$ est dans PI, d'où $g' = P(x, \_)$ et $V_1 \equiv A$ et $V_2 \equiv P^{\exists}$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(A)$ et $\Phi(\mathcal{T}) \models \Phi(\exists P \sqsubseteq A)$. Par conséquent, $\Phi(\mathcal{T}) \models A$ et $\Phi(\mathcal{T}) \models \neg P^{\exists} \vee A$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg A\} \models \neg P^{\exists}$.
3. $A(x)$ est dans $cl\_gr(g, PI)$ et $\exists P^- \sqsubseteq A$ est dans PI, d'où $g' = P(\_, x)$ et $V_1 \equiv A$ et $V_2 \equiv P^{\exists-}$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(A)$ et $\Phi(\mathcal{T}) \models \Phi(\exists P^- \sqsubseteq A)$. Par conséquent, $\Phi(\mathcal{T}) \models A$ et $\Phi(\mathcal{T}) \models \neg P^{\exists-} \vee A$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg A\} \models \neg P^{\exists-}$.
4. $P(x, y)$ est dans $cl\_gr(g, PI)$ et $Q \sqsubseteq P$ ou $Q^- \sqsubseteq P^-$ est dans PI, d'où $g' = Q(x, y)$ et $V_1 \equiv P$ et $V_2 \equiv Q$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(P)$ et $\Phi(\mathcal{T}) \models \Phi(Q \sqsubseteq P)$ (ou $\Phi(\mathcal{T}) \models \Phi(Q^- \sqsubseteq P^-)$) . Par conséquent, $\Phi(\mathcal{T}) \models P$ et $\Phi(\mathcal{T}) \models \neg Q \vee P$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P\} \models \neg Q$.
5. $P(x, y)$ est dans $cl\_gr(g, PI)$ et $Q \sqsubseteq P^-$ ou $Q^- \sqsubseteq P$ est dans PI, d'où $g' = Q(y, x)$ et $V_1 \equiv P$ et $V_2 \equiv Q^-$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(P)$ et $\Phi(\mathcal{T}) \models \Phi(Q \sqsubseteq P^-)$ (ou $\Phi(\mathcal{T}) \models \Phi(Q^- \sqsubseteq P)$) . Par conséquent, $\Phi(\mathcal{T}) \models P$ et $\Phi(\mathcal{T}) \models \neg Q^- \vee P$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P\} \models \neg Q^-$.
6. $P(x, \_)$ est dans $cl\_gr(g, PI)$ et $A \sqsubseteq \exists P$ est dans PI, d'où $g' = A(x)$ et $V_1 \equiv P^{\exists}$ et $V_2 \equiv A$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(\exists P)$ et $\Phi(\mathcal{T}) \models \Phi(A \sqsubseteq \exists P)$. Par conséquent, $\Phi(\mathcal{T}) \models P^{\exists}$ et $\Phi(\mathcal{T}) \models \neg A \vee \neg P^{\exists}$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P^{\exists}\} \models \neg A$.
7. $P(x, \_)$ est dans $cl\_gr(g, PI)$ et $\exists Q \sqsubseteq \exists P$ est dans PI, d'où $g' = Q(x, \_)$ et $V_1 \equiv P^{\exists}$ et $V_2 \equiv Q^{\exists}$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r-1$, $\Phi(\mathcal{T}) \models \Phi(\exists P)$ et $\Phi(\mathcal{T}) \models \Phi(\exists Q \sqsubseteq \exists P)$. Par conséquent, $\Phi(\mathcal{T}) \models P^{\exists}$ et $\Phi(\mathcal{T}) \models \neg Q^{\exists} \vee P^{\exists}$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P^{\exists}\} \models \neg Q^{\exists}$.
8. $P(x, \_)$ est dans $cl\_gr(g, PI)$ et $\exists Q^- \sqsubseteq \exists P$ est dans PI, d'où $g' = Q(\_, x)$ et $V_1 \equiv P^{\exists}$ et $V_2 \equiv Q^{\exists-}$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(\exists P)$ et $\Phi(\mathcal{T}) \models \Phi(\exists Q^- \sqsubseteq \exists P)$. Par conséquent, $\Phi(\mathcal{T}) \models P^{\exists}$ et $\Phi(\mathcal{T}) \models \neg Q^{\exists-} \vee P^{\exists}$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P^{\exists}\} \models \neg Q^{\exists-}$.
9. $P(\_, x)$ est dans $cl\_gr(g, PI)$ et $A \sqsubseteq \exists P^-$ est dans PI, d'où $g' = A(x)$ et $V_1 \equiv P^{\exists-}$ et $V_2 \equiv A$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r-1$, $\Phi(\mathcal{T}) \models \Phi(\exists P^-)$ et $\Phi(\mathcal{T}) \models \Phi(A \sqsubseteq \exists P^-)$. Par conséquent, $\Phi(\mathcal{T}) \models P^{\exists-}$ et $\Phi(\mathcal{T}) \models \neg A \vee P^{\exists-}$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P^{\exists-}\} \models \neg A$.
10. $P(\_, x)$ est dans $cl\_gr(g, PI)$ et $\exists Q \sqsubseteq \exists P^-$ est dans PI, d'où $g' = Q(x, \_)$ et $V_1 \equiv P^{\exists-}$ et $V_2 \equiv Q^{\exists}$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(\exists P^-)$ et $\Phi(\mathcal{T}) \models \Phi(\exists Q \sqsubseteq \exists P^-)$. Par conséquent, $\Phi(\mathcal{T}) \models P^{\exists-}$ et $\Phi(\mathcal{T}) \models \neg Q^{\exists} \vee P^{\exists-}$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P^{\exists-}\} \models \neg Q^{\exists}$.
11. $P(\_, x)$ est dans $cl\_gr(g, PI)$ et $\exists Q^- \sqsubseteq \exists P^-$ est dans PI, d'où $g' = Q(\_, x)$ et $V_1 \equiv P^{\exists-}$ et $V_2 \equiv Q^{\exists-}$ (*cf.* règles d'encodage de la Définition 5.2). Dans ce cas, à $n = r - 1$, $\Phi(\mathcal{T}) \models \Phi(\exists P^-)$ et $\Phi(\mathcal{T}) \models \Phi(\exists Q^- \sqsubseteq \exists P^-)$. Par conséquent, $\Phi(\mathcal{T}) \models P^{\exists-}$ et $\Phi(\mathcal{T}) \models \neg Q^{\exists-} \vee P^{\exists-}$ (*cf.* règles d'encodage de la Définition 5.2). Nous avons donc $\Phi(\mathcal{T}) \cup \{\neg P^{\exists-}\} \models \neg Q^{\exists-}$.

($\Leftarrow$) Notre problème revient à trouver les impliqués premiers propres de $\neg V_1$ dans $\Phi(\mathcal{T})$. Nous le prouvons par récurrence sur la plus petite taille de preuve par résolution linéaire pour produire $\Phi(\mathcal{T}) \vdash_{LR}^{\neg V_1} \neg V_2$.

Si $n = 0$ alors $V_2 = V_1 = A$ et $g = g' = A(x)$.

Supposons que l'hypothèse de récurrence est vraie jusqu'à $n < r$ et montrons qu'elle est vraie pour $n = r$. Si $\Phi(\mathcal{T}) \vdash_{LR}^{\neg V_1} \neg V_2$ après $n = r$ résolutions, alors après $n = r-1$ résolutions ($n = r-1$ applications de règles), un littéral $\Phi(\mathcal{T}) \vdash_{LR}^{\neg V_1} \neg V_3$ (un atome g'' se rajoute à $cl - gr(g, PI)$) par l'hypothèse de récurrence et une seule résolution permettra par la suite de produire $\neg V_2$ (une seule application de règle permettra de produire g') :

1. Si $V_3 = A$ et $V_2 = A'$, une résolution qui produit $\neg A'$ n'est possible que si $\neg A' \vee A \in \Phi(\mathcal{T})$. Dans ce cas, $I = A' \sqsubseteq A \in \mathcal{T}$ et puisque $g'' = A(x)$ alors $g' = A'(x)$.
2. Si $V_3 = A$ et $V_2 = P^{\exists}$, une résolution qui produit $\neg P^{\exists}$ n'est possible que si $\neg P^{\exists} \vee A \in \Phi(\mathcal{T})$. Dans ce cas, $I = \exists P \sqsubseteq A \in \mathcal{T}$ et puisque $g'' = A(x)$ alors $g' = P(x, \_)$.
3. Si $V_3 = A$ et $V_2 = P^{\exists^-}$, une résolution qui produit $\neg P^{\exists^-}$ n'est possible que si $\neg P^{\exists^-} \vee A \in \Phi(\mathcal{T})$. Dans ce cas, $I = \exists P^- \sqsubseteq A \in \mathcal{T}$ et puisque $g'' = A(x)$ alors $g' = P(\_, x)$.
4. Si $V_3 = P$ et $V_2 = Q$, une résolution qui produit $\neg Q$ n'est possible que si $\neg Q \vee P \in \Phi(\mathcal{T})$. Dans ce cas, $I = Q \sqsubseteq P \in \mathcal{T}$ et puisque $g'' = P(x, y)$ alors $g' = Q(x, y)$.
5. Si $V_3 = P$ et $V_2 = Q^-$, une résolution qui produit $\neg Q^-$ n'est possible que si $\neg Q^- \vee P \in \Phi(\mathcal{T})$. Dans ce cas, $I = Q^- \sqsubseteq P \in \mathcal{T}$ et puisque $g'' = P(x, y)$ alors $g' = Q(y, x)$.
6. Si $V_3 = P^{\exists}$ et $V_2 = A$, une résolution qui produit $\neg A$ n'est possible que si $\neg A \vee P^{\exists} \in \Phi(\mathcal{T})$. Dans ce cas, $I = A \sqsubseteq \exists P \in \mathcal{T}$ et puisque $g'' = P(x, \_)$ alors $g' = A(x)$.
7. Si $V_3 = P^{\exists}$ et $V_2 = Q^{\exists}$, une résolution qui produit $\neg Q^{\exists}$ n'est possible que si $\neg Q^{\exists} \vee P^{\exists} \in \Phi(\mathcal{T})$. Dans ce cas, $I = \exists Q \sqsubseteq \exists P \in \mathcal{T}$ et puisque $g'' = P(x, \_)$ alors $g' = Q(x, \_)$.
8. Si $V_3 = P^{\exists}$ et $V_2 = Q^{\exists^-}$, une résolution qui produit $\neg Q^{\exists^-}$ n'est possible que si $\neg Q^{\exists^-} \vee P^{\exists} \in \Phi(\mathcal{T})$. Dans ce cas, $I = \exists Q^- \sqsubseteq \exists P \in \mathcal{T}$ et puisque $g'' = P(x, \_)$ alors $g' = Q(\_, x)$.
9. Si $V_3 = P^{\exists^-}$ et $V_2 = A$, une résolution qui produit $\neg A$ n'est possible que si $\neg A \vee P^{\exists^-} \in \Phi(\mathcal{T})$. Dans ce cas, $I = A \sqsubseteq \exists P^- \in \mathcal{T}$ et puisque $g'' = P(\_, x)$ alors $g' = A(x)$.
10. Si $V_3 = P^{\exists^-}$ et $V_2 = Q^{\exists}$, une résolution qui produit $\neg Q^{\exists}$ n'est possible que si $\neg Q^{\exists} \vee P^{\exists^-} \in \Phi(\mathcal{T})$. Dans ce cas, $I = \exists Q \sqsubseteq \exists P^- \in \mathcal{T}$ et puisque $g'' = P(\_, x)$ alors $g' = Q(x, \_)$.
11. Si $V_3 = P^{\exists^-}$ et $V_2 = Q^{\exists^-}$, une résolution qui produit $\neg Q^{\exists^-}$ n'est possible que si $\neg Q^{\exists^-} \vee P^{\exists^-} \in \Phi(\mathcal{T})$. Dans ce cas, $I = \exists Q^- \sqsubseteq \exists P^- \in \mathcal{T}$ et puisque $g'' = P(\_, x)$ alors $g' = Q(\_, x)$.

□

Étant donné le Théorème 7, le calcul décentralisé de $cl\_gr(g, PI)$ est évident en utilisant DECA$_{LR}$: si g est construit en termes du vocabulaire du pair $\mathcal{P}_i$, $g' \in cl\_gr(g, PI)$ ssi $\neg V_2 \in$DECA$_{LR}{}^i(\neg V_1)$ pour les mêmes quadruplets de valeurs de g, g', V_1 et V_2 que dans le Théorème 7.

La version décentralisée de $PerfectRef$, notée $PerfectRef^i$ lorsqu'elle s'exécute sur le pair $\mathcal{P}_i$, est fournie par l'Algorithme 8. Pour chaque atome de la requête, l'algorithme calcule – de la manière décentralisée expliquée ci-dessus – l'ensemble de toutes ses reformulations, puis il produit un ensemble de reformulations de la requête originale en construisant toutes les conjonctions possibles à partir des reformulations des atomes (notées $\oslash_{i=1}^{n} cl\_gr(g_i, PI)$ à la ligne 5, où $\oslash$ est l'opérateur de distribution conjonctive). Ces reformulations sont ensuite éventuellement simplifiées en unifiant certains de leurs atomes (lignes 8 à 11) et le processus de reformulation est itéré sur les reformulations résultantes jusqu'à ce qu'aucune simplification ne soit possible (boucle générale commençant à la ligne 4).

Algorithme 8: Version decentralisée de $PerfectRef$ s'exécutant sur le pair $\mathcal{P}_i$ du PDMS $\mathcal{S}$
$PerfectRef^i(q)$
Entrée: une requête conjonctive q sur la Tbox $\mathcal{T}_i$ du pair $\mathcal{P}_i$
Sortie: une union de requêtes conjonctives sur la Tbox $\mathcal{T}$ du PDMS $\mathcal{S}$
(1) $PR := \{q\}$
(2) $PR' := PR$
(3) **while** $PR' \neq \emptyset$
(4) (a) **foreach** $q' = g_1 \wedge g_2 \wedge \ldots \wedge g_n \in PR'$
(5) $PR'' := \mathbb{\bigcirc}_{i=1}^{n} cl\_gr(g_i, PI)$
(6) $PR' := \emptyset$
(7) (b) **foreach** $q'' \in PR''$
(8) **foreach** $g'_1, g'_2 \in q''$
(9) **if** g'_1 et g'_2 sont unifiables
(10) $PR' := PR' \cup \{\tau(reduce(q'', g'_1, g'_2))\}$
(11) $PR := PR \cup PR' \cup PR''$
(12) **return** PR

Le théorème suivant établit la correction de l'algorithme décentralisé de reformulation de requêtes $PerfectRef^i$.

Theorème 8 (Correction de $PerfectRef^i$) *Soit $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$ une Tbox d'un PDMS. Soit q une requête conjonctive sur $\mathcal{T}_i$. $PerfectRef^i(q)$ retourne le même ensemble de requêtes conjonctives que $PerfectRef(q, \mathcal{T})$.*

Sa preuve résulte (1) de l'observation que la version centralisée de $PerfectRef^i$ (dans laquelle $cl\_gr(g_i, PI)$ est calculée en itérant des applications de PI sur chaque atome g_i de la requête) produit le même résultat que le $PerfectRef$ original, et (2) du Théorème 7 et de la complétude de DECA$_{LR}$ (Théorème 1 du Chapitre 2), garantissant le calcul décentralisé de tout l'ensemble $cl\_gr(g_i, PI)$.

PDMS DL-LITE$_\mathcal{R}$ avec rôles vides (Suite) Reprenons la discussion entamée dans la Section 5.3, concernant les PDMS sans rôles vides. Avec l'algorithme décentralisé $PerfectRef^i$ de refomulation de requêtes, nous pouvons maintenant lever cette limitation. En effet, il est montré dans [6] comment vérifier la consistance d'une BC DL-LITE$_\mathcal{R}$ par reformulation via $PerfectRef$ (*cf.* Section 4.4).

Grâce au résultat du Théorème 8, nous proposons donc une nouvelle version de l'algorithme $Consistent^i$ (Algorithme 9) qui vérifie la consistance par reformulation dans un PDMS DL-LITE$_\mathcal{R}$, dans toute leur généralité.

Contrairement à l'ancienne version de $Consistent^i$, qui vérifiait la consistance à partir du calcul de la clôture des NI des pairs – ce calcul étant incomplet via notre encodage propositionnel, en présence de rôles vides –, la nouvelle version reformule les recherches de contre-exemples aux NI des pairs – ce calcul étant complet via notre encodage propositionnel, même en présence de rôles vides –.

Le nouveau $Consistent^i$ est obtenue à partir de l'ancien (Algorithme 7 de la section 5.3) en remplaçant :

- $\alpha \in cln(\mathcal{P}_i)$ à la ligne 2 par $NI\ \alpha \in \mathcal{T}_i$ où α est maintenant un NI de $\mathcal{T}_i$ et
- $q_{unsat} := q_{unsat} \vee \delta(\alpha)$ à la ligne 3 par $q_{unsat} := q_{unsat} \vee PerfectRef^i(\delta(\alpha))$ où $\delta(\alpha)$ est maintenant reformulée par $PerfectRef^i$ et toutes les reformulations obtenues sont ajoutées à q_{unsat}.

Algorithme 9: L'algorithme *Consistent* s'exécutant sur le pair $\mathcal{P}_i$ du PDMS $\mathcal{S}$
$Consistent^i()$
Entrée: aucune
Sortie: *vrai* si $\mathcal{S}$ est consistant par rapport à $\mathcal{P}_i$, $faux$ sinon
(1) $q_{unsat} := \bot$ (c-à-d. q_{unsat} est initialisée à $faux$)
(2) **foreach** $NI\ \alpha \in \mathcal{T}_i$
(3) $\quad q_{unsat} := q_{unsat} \vee PerfectRef^i(\delta(\alpha))$
(4) **if** $q_{unsat}^{db(\bigcup_{i=1}^{k} \mathcal{A}_i)} = \emptyset$ tel que $1, \ldots, k$ sont tous les identifiants de pairs apparaissant dans q_{unsat}
(5) $\quad$ **return** *vrai*
(6) **else**
(7) $\quad$ **return** $faux$

Le théorème suivant formalise la *levée* de la limitation des PDMS DL-LITE$_\mathcal{R}$ à être sans rôle vide.

Theorème 9 (Correction de la vérification de la consistance) *Soit* $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ *un PDMS* DL-LITE$_\mathcal{R}$. $\mathcal{S}$ *est consistant ssi* $Consistent^i$ *retourne true pour tout* $i = 1..n$.

La preuve découle directement du Théorème 6, du Théorème 8 et de [6] (*cf.* Section 4.4).

Reconsidérons le PDMS $\mathcal{S}$ de la Section 5.3 et illustrons que l'inconsistance ne pouvait être détectée par la clôture des NI, peut être maintenant détectée par la reformulation des NI avec $perfectRef^i$. En effet, $PerfectRef^1(\exists x \exists y\ R_2(x,y) \wedge R_1(x,y))$ renvoie les reformulations suivantes, parmis les reformulations retournées :

- $q_1 = \exists x \exists y\ R_2(x,y) \wedge R_1(x,y)$.
- $q_2 = \exists x \exists y\ R_1(x,y) \wedge R_1(x,y)$ par reformulation de $R_2(x,y)$ en $R_1(x,y)$ dans q_1.
- $q_3 = R_1(\_,\_)$ par unification des $R_1(x,y)$ dans q_2 et remplacement des variables existentielles (apparaissant une seule fois dans la reformulation) par _.
- $q_4 = A_1(\_)$ par reformulation de $R_1(\_,\_)$ par $A_1(\_)$ dans q_3.
- $q_5 = B_1(\_)$ par reformulation de $R_1(\_,\_)$ par $B_1(\_)$ dans q_3.

En évaluant q_4 ou q_5 sur la Abox, on obtient le tuple vide () du fait de la constante a et donc le système est inconsistant par rapport à $\mathcal{P}_1$.

5.4.2 Evaluation des reformulations

Contrairement à l'algorithme *Answer* original, un algorithme décentralisé de calcul de réponses à une requête n'est pas capable de vérifier la consistance globale du PDMS : ceci impose au pair interrogé de connaître tous les pairs du PDMS (*cf.* Théorème 9), ce qui n'est pas le cas dans un cadre complètement décentralisé. Cependant, le pair interrogé est capable de vérifier s'il est dans un sous-PDMS consistant (*cf.* section 5.3). Dans ce cas, nous proposons de fournir des *réponses bien fondées* à la requête. Plus formellement, étant donné un PDMS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ dont la BC est éventuellement inconsistante, une réponse bien fondée à une requête posée à un pair est une réponse dérivée d'un sous-PDMS $\mathcal{S}' \subseteq \mathcal{S}$ donc la BC *est* consistante.

En effet, bien que le pair interrogé ne connaît pas tous les pairs du PDMS, il peut obtenir les identifiants des pairs $\{\mathcal{P}_j\}_{j=1..k}$ participant à une reformulation de la requête (pour les contacter) grâce aux identifiants utilisés dans les noms de concepts et rôles atomiques apparaissant dans cette reformulation. Ainsi, la consistance du PDMS par rapport à chacun de ces pairs peut être

vérifiée. Si le PDMS est consistant par rapport à chacun d'eux, on peut alors se restreindre au sous-PDMS consistant $\bigcup_{j=1}^{k}(\mathcal{T}_j \cup \mathcal{A}_j)$ pour calculer les réponses bien fondées.

L'algorithme 10 décrit l'algorithme décentralisé $Answer^i$ qui vérifie de façon décentralisée si $\bigcup_{j=1}^{k}(\mathcal{T}_j \cup \mathcal{A}_j)$ est consistante et calcule les réponses bien fondées à la requête en évaluant l'union de ses reformulations sur les Aboxes pertinentes, c-à-d. $\bigcup_{j=[1..k]} \mathcal{A}_j$. Dans cet algorithme $\perp$ remplace $AllTup(Q, \mathbf{K})$ par rapport à l'algorithme $Answer$ original.

Algorithme 10: L'algorithme décentralisé $Answer$ s'exécutant sur le pair $\mathcal{P}_i$ du PDMS $\mathcal{S}$
$Answer^i(Q)$
Entrée: une union de requêtes conjonctives Q sur la BC $\mathbf{K}_i = \langle \mathcal{T}_i, \mathcal{A}_i \rangle$ de $\mathcal{P}_i$
Sortie: Un ensemble de réponses bien fondées à Q
(1) $q = \bigcup_{q' \in Q} PerfectRef^i(q')$
(2) **if** $Consistent^j()$ retourne *true* pour tout identifiant de pair j apparaissant dans q
(3) **return** $q^{db(\bigcup_{j=1}^{k} \mathcal{A}_j)}$ tel que $1, \ldots, k$ sont tous les identifiants de pairs apparaissant dans q
(4) **else return** le singleton $\{\perp\}$

Exemple. 17 (Répondre aux requêtes dans un PDMS DL-LITE$_\mathcal{R}$**)**
Considérons le PDMS de la Figure 5.1 avec $\mathcal{A}_1 = \{a_1(Algebre, Marc)\}$, $\mathcal{A}_2 = \{par_2(Pierre, Vincent)\}$, $\mathcal{A}_3 = \{ens_3(Marc, Analyse)\}$ et $\mathcal{A}_4 = \{ppar_4(CS, Pierre)\}$.

Étant donnée la requête $q(x) = ET_2(x)$ demandant au pair $\mathcal{P}_2$ de trouver tous les étudiants du PDMS, $Answer^2(q)$ calcule la reformulation $\{ET_2(x)\} \cup \{ETA_2(x)\} \cup \{a_1(\_, x)\} \cup \{par_2(x, \_)\} \cup \{ppar_4(x, \_)\}$. Grâce à la version décentralisée de $Consistent$ (*cf.* Algorithme 9), $Answer^2(q)$ sait que $\mathcal{P}_1$ contribue à l'inconsistance du système ($Consistent^1$ trouve $Marc$ dans la recherche des contre-exemples à la NI $ET_2 \sqsubseteq \neg EN_3$). $Answer^2(q)$ retourne donc $\perp$.

Étant donnée la requête $q'(x) = DIP_4(x)$ demandant au pair $\mathcal{P}_4$ de trouver les diplômes recensés dans le PDMS, $Answer^4(q')$ calcule la reformulation $\{DIP_4(x)\} \cup \{ppar_4(x, \_)\} \cup \{val_2(\_, x)\}$. Puisque le PDMS est inconsistant (*cf.* ci-dessus) mais que $\mathcal{P}_2$ et $\mathcal{P}_4$ ne contribuent pas à cette inconsistance, $Answer^4(q')$ retourne la réponse bien fondée $\{(CS)\}$. ■

5.5 Travaux connexes

La gestion décentalisée de données a donné lieu à de nombreux travaux dans la littérature. Nous donnons ici un aperçu des principaux travaux pour le modèle relationnel et pour le Web Sémantique.

5.5.1 Les PDMS fondés sur le modèle relationnel

Dans [66, 46], les auteurs introduisent la notion de système de gestion de données pair-à-pair ou PDMS. Dans le système proposé, nommé PIAZZA, chaque pair gère localement une base de données relationnelles. Deux pairs $\mathcal{P}_1$ et $\mathcal{P}_2$ peuvent être connectés grâce à des mappings de la forme $cq_1 \subseteq cq_2$ signifiant que la requête conjonctive cq_1 exprimée sur la base de données de $\mathcal{P}_1$ est incluse dans la requête conjonctive cq_2 exprimée sur la base de données de $\mathcal{P}_2$: cq_1 fournit un sous-ensemble des réponses à cq_2. Un tel PDMS, constitué des relations et mappings des pairs, est vu et interprété comme une théorie (distribuée) de la logique du premier ordre.
Le principal résultat de [66, 46] est que le problème de répondre à une requête posée à un pair,

en termes de ses propres relations, est indécidable dans le cas général. Toutefois si la topologie du PDMS, induite par les mappings, est acyclique alors le problème devient décidable et polynomial dans la taille des données.

Dans [27, 39], les auteurs revisitent l'approche Piazza afin que répondre à une requête soit toujours décidable, même en présence de cycles dans les mappings. Pour cela, un PDMS n'est plus vu comme une théorie (distribuée) de la logique du premier ordre, mais comme une théorie (distribuée) de la logique épistémique du premier ordre (fondée sur la logique modale $S5$). Toutefois, bien que cette sémantique garantisse la décidabilité de répondre à une requête (en temps polynomial dans la taille des données), quelle que soit la topologie du PDMS, une nouvelle limitation apparaît : un mapping n'est plus une règle d'intégration entre les connaissances de deux pairs, mais simple règle d'échange de connaissances entre un pair source et un pair cible : les mappings deviennent unidirectionnels.
Ces travaux sont étendus dans [24, 26], où les auteurs étudient comment répondre à une requête quand le PDMS est inconsistant. Pour cela, ils recourent à une nouvelle sémantique épistémique (en utilisant la logique multimodale non monotone $K45_m^{\mathcal{A}}$) dans le but de définir des mappings unidirectionnels qui ne propagent pas à d'autres pairs, l'insatisfiabilité locale à un pair ou globale au PDMS.

Discussion Une des caractéristiques des PDMS relationnels est la nécessité de restreindre le modèle de données afin que répondre à une requête soit décidable. Ceci est réalisé par des contraintes topologiques (acyclicité des mappings) ou par l'utilisation de sémantiques alternatives. Nos PDMS DL-lite$_{\mathcal{R}}$ ne souffrent pas de telles limitations, du fait de leur expressivité "maîtrisée".

5.5.2 Les PDMS pour le Web Sémantique

Nous avons proposé dans ce chapitre un PDMS pour le Web Sémantique basé sur le modèle de données DL-lite$_{\mathcal{R}}$, donnant lieu au profile QL d'OWL2. Cependant, deux PDMS proches de part leur conception ont été déjà proposés dans la littérature : SomeOWL [8] et SomeRDFS [9].

Le modèle de données choisi pour le PDMS SomeOWL est un fragment de la recommandation OWL DL du W3C réduit aux constructeurs de disjonction, conjonction et négation pour construire les descriptions de classes. Le fragment considéré correspond à la logique de description $\mathcal{CLU}$. Ce modèle de données permet en particulier de définir les ontologies et les mappings sous forme d'inclusions entre des combinaisons booléennes classes. Le langage de requêtes est une combinaison booléenne de classes d'un pair donné.

Le modèle de données d'un PDMS SomeRDFS se fonde sur le fragment de logique de description de la recommandation RDFS du W3C. RDFS permet de définir les ontologies et les mappings. Le langage de requêtes considéré est celui des requêtes conjonctives en termes de classes et propriétés d'un pair.

Pour répondre à des requête, la stratégie est la même dans SomeOWL et SomeRDFS : elle consiste à réduire le problème à un calcul de conséquences dans un P2PIS propositionnel. Ce calcul de conséquences est effectué par l'algorithme DeCA (*cf.* section 2.3 du chapitre 2). Les expériences ont montré le passage à l'échelle de SomeOWL et SomeRDFS pour des réseaux allant jusqu'à mille pair [7, 8, 9].

Discussion Les PDMS DL-lite$_{\mathcal{R}}$ s'inscrive dans l'approche par compilation propositionnelle introduite par SomeOWL[8] et SomeRDFS [9]. Notamment, le modèle de données des PDMS

DL-LITE$_\mathcal{R}$ étend significativement le modèle de données de SOMERDFS limité aux seules PI de la forme : $A_1 \sqsubseteq A_2$, $P_1 \sqsubseteq P_2$, $\exists P \sqsubseteq A$ et $\exists P^- \sqsubseteq A$.
D'autre part, les expérimentations effectuées sur l'encodage propositionnel de PDMS SOMERDFS constitué de clauses binaires – sur lesquelles les méthodes de déduction fondées sur la résolution sont très efficaces – montrent d'excellentes performances. Puisque (i) l'encodage propositionnel de PDMS DL-LITE$_\mathcal{R}$ est constitué de clauses binaires uniquement, et que (ii) nous raisonnons sur cette encodage grâce à DECA$_{LR}$ qui peut être vue comme une variante optimisée de DECA pour le calcul d'impliqués (premiers propres), les performances des PDMS DL-LITE$_\mathcal{R}$ devraient être au moins aussi bonnes que celles des PDMS SOMERDFS. Ceci devra bien évidemment être vérifié par des expérimentations réelles du type de [7].

Chapitre 6

Gestion de données en DL-LITE$_\mathcal{R}$ au travers de vues

Nous présentons dans ce chapitre la quatrième et dernière contribution de cette thèse : un modèle distribué de données en DL-LITE$_\mathcal{R}$ étendu par des vues et les algorithmes décentralisés associés de gestion de données.

Une vue est tout simplement une requête prédéfinie dont l'ensemble des réponses, appelées *extensions*, est disponible. Cette notion est centrale en gestion de données où les vues sont à la base de nombreux mécanismes de sécurité (les données d'une base sont volontairement cachées, seules les données des vues sont accessibles/publiées), d'optimisation (les (sous-)requêtes fréquentes sont précalculées/matérialisées) ou encore d'échange de données entre systèmes dont les schémas diffèrent (les réponses à une requête sur un système sont des tuples fournis par un autre système).

Afin de généraliser les pairs des PDMS DL-LITE$_\mathcal{R}$ à des pairs offrant des mécanismes classiques de gestion de données fondés sur les vues, nous revisitons les résultats du chapitre précédent en considérant que chaque pair ne dispose plus d'une Abox mais d'un ensemble de vues.

Notre approche consiste à combiner les algorithmes de gestion décentralisée de données présentés dans le Chapitre 5 et l'algorithme MiniCon [59] bien connu dans la littérature pour le calcul de réécritures maximales d'une requête en termes de vues.

Nous présentons l'algorithme MiniCon dans la Section 6.1. Nous décrivons ensuite le modèle (centralisé) DL-LITE$_\mathcal{R}$ étendu par des vues dans la Section 6.2. Nous abordons la gestion centralisée de données dans ce modèle dans la Section 6.3. Enfin, dans la Section 6.4, nous étendons ces résultats avec ceux du Chapitre 5 afin de gérer des données au travers de vues dans les PDMS DL-LITE$_\mathcal{R}$.

6.1 Préliminaires : l'algorithme MiniCon

L'algorithme MiniCon de [59] est l'algorithme de référence dans la littérature pour le calcul de réécritures maximales d'une requête en termes de vues.

Définition 12 (Réécriture maximale d'une requête en termes de vues) *Soient q une requête conjonctive sur un schéma de base de données relationnelles $\mathcal{S}$ et $\mathcal{V}$ un ensemble de vues conjonctives sur $\mathcal{S}$.*

- *Une* réécriture *de q est une requête conjonctive q_v dont les prédicats apparaissant dans sa définition sont des noms de vues de $\mathcal{V}$ et telle que $\mathcal{S} \cup \mathcal{V} \models \forall \bar{x}\,(q_v(\bar{x}) \Rightarrow q(\bar{x}))$.*
- *Une réécriture q_v de q est* maximale *s'il n'existe pas d'autre réécriture de q strictement plus générale que q_v.*

La propriété suivante formalise un résultat important sur le calcul de réécriture offert par MiniCon.

Propriété 8

- *MiniCon est correct pour le calcul de réécriture.*
- *MiniCon est complet pour le calcul de réécriture maximale.*

Cette propriété a des conséquences très importantes, puisqu'il a été montré ([45]) que l'ensemble des réponses (certaines) à une requête peut être obtenu en évaluant sur les extensions de vues l'union (finie) de ses réécritures maximales. C'est précisément ce que permet de faire MiniCon.

6.2 Modèle de données DL-LITE$_\mathcal{R}$ avec des vues

Nous définissons une *vue* v par une requête conjonctive $v(\bar{x}) = \exists \bar{y}\ conj(\bar{x}, \bar{y})$ ayant une extension $\mathcal{E}(v)$ qui est un ensemble de faits de la forme $v(\bar{t})$. En suivant l'hypothèse du monde ouvert – classique en intégration d'information –, nous supposons que nos vues sont *correctes*, c-à-d. pour toute interprétation I et pour tout $v(\bar{t}) \in \mathcal{E}(v)$, $\bar{t}^I \in v^I$: l'extension d'une vue ne fournit que des réponses à cette vue, mais pas nécessairement toutes.

Une BC $\mathbf{K}$ est maintenant de la forme $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$, où $\mathcal{E}$ est un ensemble de faits de la forme $v(\bar{t})$ et où v est une *vue* de $\mathcal{V}$ utilisant les concepts et rôles atomiques de $\mathcal{T}$.
Un *modèle* de $\mathbf{K}$ est une interprétation I qui est à la fois un modèle de $\mathcal{T}$, $\mathcal{V}$ et $\mathcal{E}$. $\mathbf{K}$ est *consistante* si elle a au moins un modèle. Enfin, $\mathbf{K}$ *implique logiquement* une assertion β (PI, NI, ou fait), noté $\mathbf{K} \models \beta$, si tout modèle de $\mathbf{K}$ est un modèle de β.

En présence de vues, les *réponses certaines* à des requêtes sont celles pouvant être inférées à partir des extensions de vues.
L'ensemble des réponses certaines à une requête n-aire non booléenne q sur $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ est : $cert(q, \mathbf{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathbf{K} \models q(\bar{t})\}$, où $\mathcal{C}$ est l'ensemble des constantes de $\mathcal{E}$ et $q(\bar{t})$ est la formule *close* obtenue en remplaçant les variables libres de $\bar{x}$ par les constantes de $\bar{t}$ dans la définition de la requête. Par convention, l'ensemble des réponses certaines à une requête booléenne est $\{()\}$, où $()$ est le tuple vide, si $\mathbf{K} \models q()$, et $\emptyset$ sinon.

6.3 Gestion *centralisée* de données au travers de vues

Nous traitons dans cette section la gestion centralisée de données en termes de vues dans DL-LITE$_\mathcal{R}$. Nous abordons tout d'abord le problème de vérification de la consistance puis le problème de répondre à des requêtes.

6.3.1 Vérification de la consistance

Dans la vérification de la consistance d'une BC DL-LITE$_\mathcal{R}$ $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ sans vues (*cf.* Section 4.2 du Chapitre 4), on calcule tout d'abord la clôture des NI $cln(\mathcal{T})$ qui est ensuite transformée

en une union de requêtes booléennes Q modélisant la recherche des contre-exemples à ces NI. L'évaluation de Q sur $\mathcal{A}$ – considérée comme une base de données relationnelles $db(\mathcal{A})$ – permet de décider de la consistance de $\mathbf{K}$.

Considérons maintenant une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. Étant donné que les données ne sont plus représentées par des faits dans $\mathcal{A}$, mais par des extensions $\mathcal{E}$ d'un ensemble de vues $\mathcal{V}$ définies sur $\mathcal{T}$, nous réécrivons tout d'abord Q en termes de vues via l'algorithme MiniCon et nous évaluons ensuite les réécritures de Q sur $\mathcal{E}$ – considérée comme une base de données relationnelles $db(\mathcal{E})$ – afin de décider de la consistance de $\mathbf{K}$.

L'algorithme $ViewConsistent$ (Algorithme 11) est la variante de l'algorithme $Consistent$ original (*cf.* Algorithme 3 du Chapitre 4) obtenue à partir de l'algorithme $Consistent$ original en remplaçant :

- $q_{unsat} := q_{unsat} \cup \{\delta(\alpha)\}$ à la ligne 3 de l'Algorithme 3 par $q_{unsat} := q_{unsat} \cup MiniCon(\delta(\alpha), \mathcal{V})$, où $MiniCon(\delta(\alpha), \mathcal{V})$ fournit les réécritures maximales en termes de $\mathcal{V}$ de la traduction en logique du premier ordre $\delta(\alpha)$ des NI α impliquées logiquement par $\mathcal{T}$,
- $q_{unsat}^{db(\mathcal{A})}$ à la ligne (4) de l'Algorithme 3 par $q_{unsat}^{db(\mathcal{E})}$, c-à-d. l'évaluation de q_{unsat} – au sens des bases de données relationnelles – sur les extensions de vues.

Algorithme 11: L'algorithme $ViewConsistent$
$ViewConsistent(\mathbf{K})$
Entrée: une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$
Sortie: $vrai$ si $\mathbf{K}$ est satisfiable, $faux$ sinon
(1) $q_{unsat} := \bot$ (c-à-d. $q_{unsat} = faux$)
(2) **foreach** $\alpha \in cln(\mathcal{T})$
(3) $\quad q_{unsat} := q_{unsat} \vee MiniCon(\delta(\alpha), \mathcal{V})$
(4) **if** $q_{unsat}^{db(\mathcal{E})} = \emptyset$
(5) $\quad$ **return** $vrai$
(6) **else**
(7) $\quad$ **return** $faux$

Theorème 10 (Correction de la vérification de consistance) *Soit* $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ *une BC.* $\mathbf{K}$ *est consistante ssi* $ViewConsistent(\mathbf{K})$ *retourne vrai.*

Considérons de façon équivalente que $\mathbf{K}$ est inconsistante ssi $ViewConsistent(\mathbf{K})$ retourne $faux$.

De $\alpha \in cln(\mathcal{T})$ et $q_v \in MiniCon(\delta(\alpha), \mathcal{V})$ nous déduisons $\mathcal{T} \cup \mathcal{V} \models (q_v \Rightarrow \delta(\alpha))$ et ainsi q_v est une réécriture conjonctive de $\delta(\alpha)$, dont l'évaluation fournit des réponses certaines. Il en découle que $\mathbf{K}$ est inconsistante dès lors que $ViewConsistent(\mathbf{K})$ retourne $false$.

Observons maintenant que $\mathcal{T} \cup \mathcal{V}$ et $\mathcal{V} \cup \mathcal{E}$ sont toujours satisfiables. Donc, si $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ est inconsistante alors $\alpha \in cln(\mathcal{T})$ telle que $\delta(\alpha) = \exists \bar{x}\ q(\bar{x})$ et $\mathcal{V} \cup \mathcal{E} \models \exists \bar{x}\ q(\bar{x})$. Soit $\bar{t}$ un tuple fait de constantes apparaissant dans $\mathcal{E}$ tel que $\mathcal{V} \cup \mathcal{E} \models q(\bar{t})$. En réutilisant la notion de témoin d'un tuple introduite dans le Lemme 39 de [25] sur $\mathcal{E}$ au lieu de $\mathcal{A}$, nous construisons à partir du témoin de $\bar{t}$ une requête conjonctive spécifique q_v en termes de vues telle que $\bar{t}$ est dans son ensemble de réponses et nous montrons que cette requête est une réécriture de q. Puisque $MiniCon(\delta(\alpha), \mathcal{V})$ calcule toutes les réécritures maximales de $\delta(\alpha)$ (Théorème 1 de [59]), q_v spécialise l'une d'elles, donc $ViewConsistent(\mathbf{K})$ retourne $false$.

Exemple. 18 (Vérification de la consistance en termes de vues)
La Figure 6.1 illustre la Tbox $\mathcal{T}$ d'un service de cours particuliers $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. Les concepts sont notés en majuscule et les rôles en minuscule.

Le service gère (i) des cours (C) de langues (L) ou de maths (M) ($L \sqsubseteq C$, $M \sqsubseteq C$), (ii) des personnes (P) étudiants (ET) ou enseignants (EN) ($ET \sqsubseteq P$, $EN \sqsubseteq P$) et habitant dans des villes (V) ($\exists pv \sqsubseteq P$ et $\exists pv^- \sqsubseteq V$). Les étudiants sont inscrits à des cours ($\exists ins \sqsubseteq ET$ et $\exists ins^- \sqsubseteq C$) et les enseignants sont responsables de cours ($\exists res \sqsubseteq EN$ et $\exists res^- \sqsubseteq C$).

$ET \sqsubseteq P$	$L \sqsubseteq C$
$EN \sqsubseteq P$	$M \sqsubseteq C$
$\exists pv \sqsubseteq P$	$\exists pv^- \sqsubseteq V$
$\exists res \sqsubseteq EN$	$\exists res^- \sqsubseteq C$
$\exists ins \sqsubseteq ET$	$\exists ins^- \sqsubseteq C$

FIGURE 6.1 – La Tbox $\mathcal{T}$ de $\mathbf{K}$

L'ensemble des vues $\mathcal{V}$ définies sur $\mathbf{K}$ est le suivant :
- $v_1(x, y) = ET(x) \wedge pv(x, y)$ décrivant les étudiants et les villes où ils habitent.
- $v_2(x, y) = EN(x) \wedge pv(x, y)$ décrivant les enseignants et les villes où ils habitent.
- $v_3(x) = \exists y\ res(x, y) \wedge M(y)$ décrivant les enseignants responsables de cours de maths.
- $v_4(x) = \exists y\ ins(x, y) \wedge L(y)$ décrivant les étudiants inscrits aux cours de langues.
- $v_5(x, y) = \exists z\ res(x, z) \wedge ins(y, z)$ décrivant des étudiants inscrits dans un cours et les enseignants responsables du même cours.

Supposons que $ET \sqsubseteq \neg EN \in \mathcal{T}$, c-à-d. les étudiants et les enseignants sont disjoints, et que $\mathcal{E}$ = $\{v_1(Jean, Paris)$, $v_1(Vincent, Orsay)$, $v_2(Jean, Rennes)$, $v_5(Pierre, Jean)\}$.

Pour vérifier la consistance de $\mathbf{K}$, $ViewConsistent$ calcule tout d'abord $cln(\mathcal{T})$. Celle-ci contient la NI $ET \sqsubseteq \neg EN$ dont la requête de recherche de contre-exemple est $\delta(ET \sqsubseteq \neg EN) = \exists x\ ET(x) \wedge EN(x)$. Parmi ses réécritures renvoyées par $MiniCon(\exists x\ ET(x) \wedge EN(x), \mathcal{V})$ on trouve la réécriture $\{\exists x \exists y_1\ v_1(x, y_1) \wedge \exists y_2\ v_2(x, y_2)\}$. Cette réécriture est ajoutée alors à q_{unsat} à la ligne 3 de l'algorithme 11 et $q_{unsat}^{db(\mathcal{E})} \neq \emptyset$ à la ligne 4. Ceci implique que $\mathbf{K}$ est inconsistante.

Supposons maintenant que $\mathcal{E} = \{v_1(Vincent, Orsay)$,$v_1(Paul, Rennes)$, $v_2(Jean, Rennes)$, $v_5(Pierre, Jean)\}$. Aucune réécriture en termes de vues d'une recherche de contre-exemple à une NI ne possède de réponses dans $db(\mathcal{E})$. $\mathbf{K}$ est donc consistante. ■

6.3.2 Répondre à des requêtes

Pour répondre à des requêtes dans une BC DL-LITE$_\mathcal{R}$ sans vues $\mathbf{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (*cf.* Section 4.3 du Chapitre 4), après le test de consistance, on procéde tout d'abord par reformulation de requête à l'aide de l'algorithme $PerfectRef$ puis par évaluation des reformulations sur $\mathcal{A}$, considérée comme une base de données relationnelles $db(\mathcal{A})$.

Considérons maintenant la BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. La vérification de la consistance est réalisée par l'algorithme $ViewConsistent$ (*cf.* Section 6.3.1). Ensuite, les données étant représentées par des extensions de vues, les reformulations obtenues par $PerfectRef$ doivent être réécrites en termes vues avant d'être évaluées sur $\mathcal{E}$, considérée comme une base de données relationnelles $db(\mathcal{E})$.

CertainAnswer (Algorithme 12) est une variante de l'algorithme *Answer* original qui calcule les réponses certaines d'une union de requêtes conjonctives Q sur une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. Si la BC est inconsistante, l'algorithme retourne $Alltup(Q, \mathcal{E})$, l'ensemble de tous les tuples d'arité de Q pouvant être générés à partir des constantes de $\mathcal{E}$. Sinon, il calcule (en utilisant $MiniCon(q', \mathcal{V})$) puis évalue les réécritures des requêtes conjonctives q' retournées par $PerfectRef$ sur $\mathcal{E}$.

Algorithme 12: L'algorithme *CertainAnswer*
$CertainAnswer(Q, \mathbf{K})$
Entrée: une union de requêtes conjonctives Q et une BC $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$
Sortie: $cert(Q, \mathbf{K})$
(1) **if** not $ViewConsistent(\mathbf{K})$
(2) **return** $Alltup(Q, \mathcal{E})$
(3) **else**
(4) $Q' := \bigcup_{q \in Q} PerfectRef(q, \mathcal{T})$
(5) **return** $(\bigcup_{q' \in Q'} MiniCon(q', \mathcal{V}))^{db(\mathcal{E})}$

Theorème 11 (Correction de *CertainAnswer***)** *Soient* $\mathbf{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ *une BC et* Q *une union de requêtes conjonctives.* $cert(Q, \mathbf{K}) = CertainAnswer(Q, \mathbf{K})$.

Quand $\mathbf{K}$ est inconsistante, la correction de *CertainAnswer* découle directement de celle de *ViewConsistent*.
Considérons maintenant que $\mathbf{K}$ est consistante. De $q' \in PerfectRef(q, \mathcal{T})$ et $q_v \in MiniCon(q', \mathcal{V})$ nous déduisons $\mathcal{T} \cup \mathcal{V} \models \forall \bar{x} \, (q_v(\bar{x}) \Rightarrow q(\bar{x}))$ et ainsi q_v est une réécriture conjonctive de q, dont l'évaluation fournit des réponses certaines.
Maintenant, si $\bar{t}$ est une réponse certaine, en réutilisant la notion de témoin d'un tuple introduite dans le Lemme 39 de [25] sur $\mathcal{E}$ au lieu de $\mathcal{A}$, nous construisons à partir du témoin de $\bar{t}$ une requête conjonctive spécifique q_v en termes de vues telle que $\bar{t}$ est dans son ensemble de réponses et nous montrons par récurrence que cette requête est une réécriture d'une reformulation q' de q. Puisque *MiniCon* calcule toutes les réécritures maximales de q' (Théorème 1 de [59]), soit q_v est l'une d'elles, soit q_v soit implique logiquement l'une d'elles, et $\bar{t}$ sera donc retourné par la ligne 5 de l'Algorithme 12.

Exemple. 19 (Répondre à des requêtes en termes de vues)
Reconsidérons l'exemple de la section précédente et retirons la NI $ET \sqsubseteq \neg EN$ que nous avons ajouté à $\mathcal{T}$ pour la vérification de la consistance.

Supposons que $\mathcal{E} = \{v_1(Vincent, Orsay), v_1(Paul, Orsay), v_4(Paul), \; v_5(Paul, Jean)\}$ et calculons les réponses à la requête $q(x, y) = \exists v \; ET(x) \wedge pv(x, v) \wedge EN(y) \wedge pv(y, v)$ demandant de trouver les couples étudiant/enseignant habitant dans la même ville.

Tout d'abord $\mathbf{K}$ est consistante car il n'y a aucune NI dans $\mathcal{T}$. *CertainAnswer* calcule donc les réécritures de q à l'aide de $PerfectRef$. Parmi ces réécritures on trouve :
- $q_1(x, y) = \exists v \; ET(x) \wedge pv(x, v) \wedge EN(y) \wedge pv(y, v)$ (la requête posée),
- $q_2(x, x) = ET(x) \wedge pv(x, \_) \wedge EN(x)$ (par unification des deux atomes pv et remplacement de la variable existentielle v par _,
- $q_3(x, x) = ins(x, \_) \wedge pv(x, \_) \wedge EN(x)$ (par reformulation de $ET(x)$ par $ins(x, \_)$ grâce à la PI $\exists ins \sqsubseteq ET$ de $\mathcal{T}$),

- $q_4(x,x) = ET(x) \wedge pv(x,\_) \wedge res(x,\_)$ (par reformulation de $EN(x)$ par $res(x,\_)$ grâce à la PI $\exists res \sqsubseteq EN$ de $\mathcal{T}$),
- $q_5(x,x) = ins(x,\_) \wedge pv(x,\_) \wedge res(x,\_)$ (par reformulation de $ET(x)$ et $EN(x)$ de la deuxième réécritures en $ins(x,\_)$ et $res(x,\_)$,
- ...

Une fois que les reformulations sont calculées, MiniCon les réécrit en termes de vues. Prenons par exemple la réécriture $q_5(x,x) = ins(x,\_) \wedge pv(x,\_) \wedge res(x,\_)$. Une des réécritures de q_5 produite par Minicon est $q_{5v}(x,y) = v_4(x) \wedge v_1(x,x_2) \wedge v_5(x,x_3)$. L'évaluation de q_{5v} sur $db(\mathcal{E})$ produit la réponse (Paul,Paul). ■

6.4 Gestion *décentralisée* de données DL-LITE$_\mathcal{R}$ au travers de vues

Un PDMS DL-LITE$_\mathcal{R}$ $\mathcal{S}$ avec des vues est un ensemble de pairs $\{\mathcal{P}_i\}_{i=1..n}$ où chaque pair $\mathcal{P}_i$ gère sa propre BC DL-LITE$_\mathcal{R}$ $\mathbf{K}_i = \langle \mathcal{T}_i, \mathcal{V}_i, \mathcal{E}_i \rangle$. Nous notons par A_i (resp. P_i) le concept atomique A (resp. le rôle atomique P) de $\mathcal{P}_i$ et par v_i la vue v de $\mathcal{P}_i$.

Nous nous intéressons dans cette section aux problèmes de vérification de la consistance et de répondre à des requêtes au travers de vues dans les PDMS DL-LITE$_\mathcal{R}$.

6.4.1 Vérification de la consistance

Nous rappelons tout d'abord que la vérification de la consistance dans un PDMS DL-LITE$_\mathcal{R}$ sans vues s'effectue par rapport à un pair (*cf.* Section 5.3 du Chapitre 5). L'algorithme $Consistent^i$ (*cf.* Algorithme 9 du Chapitre 5) considère tout d'abord les NI du pair et calcule l'union des requêtes conjonctives booléennes Q modélisant la recherche des contre-exemples à ces NI. Ensuite, il reformule Q à l'aide de $PerfectRef^i$ et évalue les reformulations sur l'union des Aboxes $\mathcal{A}_i$ des pairs apparaissant dans une réécriture. Le résultat de cette évaluation permet de décider de la consistance du PDMS par rapport à ce pair.

Considérons maintenant un PDMS DL-LITE$_\mathcal{R}$ avec des vues. Nous vérifions la consistance du PDMS toujours par rapport à un pair (un pair ne peut pas vérifier la consistance globale du PDMS). Les données étant dans ce cas des extensions de vues, nous réécrivons tout d'abord les reformulations retournées par $PerfectRef^i$ en termes de vues à l'aide de MiniCon. Les réécritures calculées par MiniCon sont ensuite évaluées sur l'union des $\mathcal{E}_i$ des pairs $\mathcal{P}_i$ apparaissant dans une réécriture. Le résultat de cette évaluation permet de décider de la consistance du PDMS par rapport à ce pair.

Notons que pour fournir à MiniCon(q, $\mathcal{V}$) l'ensemble des vues concernées par la réécriture en termes de vues, nous définissons la fonction $fetchViews(q)$, où q est une requête conjonctive pouvant mettre en jeu le vocabulaire de différents pair : $fetchViews(q)$ récupère sur ces pairs l'ensemble des vues dont l'un des atomes peut être unifié avec un atome de la requête. Pour cela, $fetchViews(q)$ contacte les pairs dont les identifiants apparaissent dans les noms de concepts et rôles atomiques utilisés dans q.

La version décentralisée de $ViewConsistent$ est notée $ViewConsistent^i$ lorsqu'elle s'exécute sur un pair $\mathcal{P}_i$. L'algorithme $ViewConsistent^i$ (Algorithme 13) étend $Consistent^i$ en remplaçant l'évaluation de q_{unsat} sur les Aboxes pertinentes par l'évaluation de Q_{unsat} sur les extensions de vues pertinentes, où Q_{unsat} est obtenue à partir de q_{unsat} (qui est calculée comme dans $Consistent^i$) comme suit : $Q_{unsat} := \bigcup_{q \in q_{unsat}} MiniCon(q, fetchViews(q))$.

Algorithme 13: L'algorithme *Consistent* s'exécutant sur le pair $\mathcal{P}_i$ du PDMS $\mathcal{S}$
$ViewConsistent^i()$
Entrée: aucune
Sortie: *vrai* si $\mathcal{S}$ est consistant par rapport à $\mathcal{P}_i$, $faux$ sinon
(1) $q_{unsat} := \bot$ (c-à-d. q_{unsat} est initialisée à $faux$)
(2) **foreach** $NI\alpha \in \mathcal{T}_i$
(3) $q_{unsat} := q_{unsat} \vee PerfectRef^i(\delta(\alpha))$
(4) $Q_{unsat} := \bigcup_{q \in q_{unsat}} MiniCon(q, fetchViews(q))$
(5) **if** $Q_{unsat}^{db(\bigcup_{i=1}^{k} \mathcal{E}_i)} = \emptyset$ tel que $1, \ldots, k$ sont tous les identifiants de pairs apparaissant dans Q_{unsat}
(6) **return** *vrai*
(7) **else**
(8) **return** $faux$

Theorème 12 (Correction de la vérification de la consistance) *Soit $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ un PDMS avec des vues. $\mathcal{S}$ est consistant ssi $ViewConsistent^i$ retourne true pour tout $i = 1..n$.*

La preuve découle directement du Théorème 8, du Théorème 9, du Théorème 10 et de la fonction $fetchViews$.

Exemple. 20 (Service décentralisé de cours particuliers avec des vues) La Figure 6.2 représente le PDMS $\mathcal{S}$ obtenu par distribution de la BC centralisée de l'Exemple 18 de la Section précédente sur les deux pairs $\mathcal{P}_1$ et $\mathcal{P}_2$, où $\mathcal{P}_1$ gère le personnel (étudiants et enseignants) et $\mathcal{P}_2$ les cours. Les BC des pairs sont les nœuds étiquetés par les noms des pairs et les mappings entre les pairs étiquettent l'arête qui lie leur BC respective.

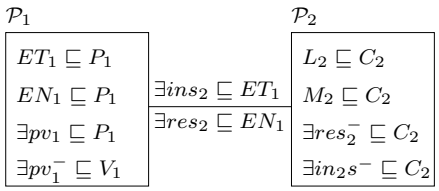

FIGURE 6.2 – Le PDMS $\mathcal{S}$

Les vues de $\mathcal{P}_1$ sont définies par :
- $a_1(x, y) = ET_1(x) \wedge pv_1(x, y)$ décrivant les étudiants de $\mathcal{P}_1$ et les villes où ils habitent.
- $b_1(x, y) = EN_1(x) \wedge pv_1(x, y)$ décrivant les enseignants $\mathcal{P}_1$ et les villes où ils habitent.

Et les vues de $\mathcal{P}_2$ sont définies par :
- $a_2(x) = \exists y\ res_2(x, y) \wedge M_2(y)$ décrivant les enseignants de $\mathcal{P}_2$ responsables de cours de maths.
- $b_2(x) = \exists y\ ins_2(x, y) \wedge L_2(y)$ décrivant les étudiants de $\mathcal{P}_2$ inscrits aux cours de langues.
- $c_2(x, y) = \exists z\ res_2(x, z) \wedge ins_2(y, z)$ décrivant des étudiants inscrits dans un cours et les enseignants responsables du même cours dans $\mathcal{P}_2$.

Supposons que la NI $ET_1 \sqsubseteq \neg EN_1 \in \mathcal{T}_1$ et que $\mathcal{E}_1 = \{a_1(Jean, Paris),\ a_1(Vincent, Orsay),\ b_1(Jean, Rennes)\}$ et $\mathcal{E}_2 = \{b_2(Jean),\ c_2(Pierre, Jean)\}$.

Pour tester la consistance de $\mathcal{S}$ par rapport à $\mathcal{P}_1$, l'algorithme $ViewConsistent^i$ considère la seule NI $ET_1 \sqsubseteq \neg EN_1$ de $\mathcal{T}_1$, calcule la requête booléenne modélisant ses éventuels contre-exemples $q = \delta(ET_1 \sqsubseteq \neg EN_1)$ et reformule q sur $\mathcal{S}$ à l'aide de $PerfectRef^1(q)$. Parmi les réécritures retournées par $PerfectRef^1(q)$, on trouve $q' = \exists x ins_2(x, \_) \wedge EN_1(x)$. q' sera donc réécrite en termes de vues à l'aide de MiniCon. Pour cela, $fetchViews(q')$ récupère les vues b_1,b_2 et c_2 et les fournit à Minicon. Parmi les réécritures produites par $MiniCon(q', \{b_1, b_2, c_2\})$, on trouve $\exists x\ b_1(x, y) \wedge b_2(x)$. $q_{unsat}^{db(\mathcal{E}_1 \cup \mathcal{E}_2)} \neq \emptyset$ à la ligne 5 et **K** est donc inconsistante par rapport à $\mathcal{P}_1$. ■

6.4.2 Répondre à des requêtes

Répondre à une requête conjonctive q posé à un pair $\mathcal{P}_i$ d'un PDMS DL-LITE$_\mathcal{R}$ sans vues $\mathcal{S}$ consiste à reformuler en premier lieu la requête q à l'aide de $PerfectRef^i$. La consistance du PDMS par rapport à tous les pairs apparaissant dans une reformulation est ensuite vérifiée. Si le PDMS est consistant par rapport à tous ces pairs, $Answer^i$ renvoie des réponses bien fondées à la requête puisqu'elles sont calculées à partir d'un sous-PDMS consistant. Sinon, $Answer^i$ renvoie $\perp$.

Résoudre le même problème dans un PDMS DL-LITE$_\mathcal{R}$ avec vues suit le même principe sauf qu'ici, la consistance du PDMS par rapport aux pairs se fait via l'algorithme $ViewConsistent^i$ et puisque les données sont des extensions de vues, nous sommes obligés de passer par une étape de réécritures en termes de vues et ceci pour l'ensemble des reformulations produites par $PerfectRef^i$. Nous utilisons donc de nouveau la fonction $fetchViews$ pour récupérer l'ensemble de vues $\mathcal{V}$ concernées par les reformulations et nous recourons à $Minicon$ pour le calcul de réécritures en termes de vues. Nous évaluons enfin les réécritures produites par MiniCon sur l'union des $\mathcal{E}_i$ des pairs $\mathcal{P}_i$ dont un atome apparaît dans une réécriture. Le résultat de cette évaluation fournit aussi des réponses bien fondées à la requête.

La version décentralisée de $CertainAnswer$ est notée $CertainAnswer^i$ lorsqu'elle s'exécute sur un pair $\mathcal{P}_j$. L'algorithme $CertainAnswer^i$ (Algorithme 14) étend $Answer^i$ en remplaçant l'évaluation sur les Aboxes pertinentes de l'union q des reformulations de la requête initiale (ligne 2 dans l'Algorithme 10) par l'évaluation de la requête $Q' := \bigcup_{q' \in q} MiniCon(q', fetchViews(q'))$ sur les extensions de vues pertinentes. L'intérêt de l'Algorithme 14 est de fournir des réponses *bien fondées*, c-à-d. des réponses dérivées à partir d'un sous-ensemble consistant de la BC (éventuellement inconsistante) du PDMS.

Algorithme 14: L'algorithme $CertainAnswer$ s'exécutant sur le pair $\mathcal{P}_i$ du PDMS $\mathcal{S}$
$CertainAnswer^i(Q)$
Entrée: une union de requêtes conjonctives Q sur la BC $\mathbf{K}_i = \langle \mathcal{T}_i, \mathcal{V}_i, \mathcal{E}_i \rangle$ de $\mathcal{P}_i$
Sortie: Un ensemble de réponses bien fondées à Q
(1) $q := \bigcup_{q' \in Q} PerfectRef^i(q')$
(2) **if** $ViewConsistent^j()$ retourne *vrai* pour tout identifiant de pair j apparaissant dans q
(3) $\quad Q' := \bigcup_{q' \in q} MiniCon(q', fetchViews(q'))$
(4) $\quad$ **return** $Q'^{db(\bigcup_{j=1}^{k} \mathcal{E}_j)}$ tel que $1, \ldots, k$ sont tous les identifiants de pairs apparaissant dans Q'
(5) **else**
(6) $\quad$ **return** le singleton $\{\perp\}$

Exemple. 21 (Répondre à des requêtes en termes de vues)
Reconsidérons le PDMS de la section précédente et retirons la NI $ET_1 \sqsubseteq \neg EN_1$ que nous avons ajouté pour la vérification de la consistance.

Supposons que $\mathcal{E}_1 = \{a_1(Vincent, Orsay), a_1(Paul, Orsay)\}$ et $\mathcal{E}_2 = \{b_2(Paul), c_2(Paul, Jean)\}$ et calculons les réponses à la requête $q(x, y) = \exists v\ ET_1(x) \wedge pv_1(x, v) \wedge EN_1(y) \wedge pv_1(y, v)$ posée à $\mathcal{P}_1$ et demandant de trouver les couples étudiant/enseignant habitant la même ville.

Tout d'abord **K** est consistante par rapport à $\mathcal{P}_1$ et $\mathcal{P}_2$ car il n'y a aucune NI dans les Tbox des pairs. $PerfectRef^1(q)$ produit les réécritures de q parmi lesquelles on trouve $q_5(x, x) = ins_2(x, \_) \wedge pv_1(x, \_) \wedge res_2(x, \_)$. $fetchViews(q_5) = \mathcal{V} = \{a_1, b_1, a_2, b_2, c_2\}$ et $Minicon(q_5, \mathcal{V})$ retourne les réécritures maximales de q_5 en termes des vues, parmi lesquelles on trouve $q_{5v}(x, x) = b_2(x) \wedge a_1(x, x_2) \wedge c_2(x, x_3)$ dont l'évaluation produit la réponse bien fondée $(Paul, Paul)$. ■

Conclusion et Perspectives

Dans cette thèse, nous avons fourni les fondements pour des raisonnements standard et non-standard dans les systèmes de gestion de données et de connaissances.

En logique propositionnelle

Nous avons tout d'abord proposé *le* premier algorithme de calcul décentralisé de conséquences par déduction linéaire dans les P2PIS propositionnels : DECA$_{LR}$. Notamment, cette contribution inclus, à notre connaissance, *la* première étude de complexité (communicationnelle, en espace et en temps) de la littérature pour un algorithme décentralisé de déduction.

Nous avons considéré la décentralisation de la plus générale des stratégie de résolution linéraire pour la déduction. Une perspective est donc d'étudier s'il est possible de décentraliser d'autres stratégies linéraires, connues comme étant complètes pour le calcul de conséquences dans le cadre centralisé. Par exemple, des stratégies linéaires fondées sur l'ordonnancement des littéraux, l'opérateur de "Skip", etc (*cf.* [55]). Décentraliser de telles stratégies serait utile en termes d'optimisation, car elles permettraient de calculer encore plus efficacement les impliqués premiers propres d'une clause dans un P2PIS.

Nous avons ensuite proposé *la* première étude de la notion d'extension (non) conservative dans un cadre logique distribué. Cette notion est importante dans les P2PIS du fait qu'elle 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 étudié théoriquement et algorithmiquement les deux 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. Pour cela, nous avons exhibé le lien entre ces problèmes et la déduction linéaire décentralisée, et en particulier comment les résoudre avec DECA$_{LR}$.

Une perspective serait de développer la notion de stratégie auto-guérissante décentralisée introduite dans cette thèse. En effet, nous ne proposons pour le moment qu'une stratégie naïve de restauration (totale ou partielle) de conservativité d'un pair : retirer les mappings incriminés. Il serait intéressant d'envisager d'autres stratégies plus élaborées. Par exemple, on pourrait supprimer les sous-ensembles minimaux de mappings permettant de retrouver la conservativité totale/partielle désirée, et éventuellement affecter un taux de confiance ou une priorité sur les mappings afin d'ordonner les différents ensembles minimaux entre eux.

En logique de description

Nous avons considéré la logique de description DL-LITE$_{\mathcal{R}}$ qui est un fragment de la recommandation OWL du W3C pour le Web Sémantique, ainsi que de sa récente évolution OWL2 : le

profile QL de OWL2 se fonde sur DL-LITE$_\mathcal{R}$.
Nous avons proposé un modèle distribué de données – éventuellement avec des vues – et les algorithmes associés de gestion de données afin de mettre en œuvre des PDMS pour le Web sémantique fondés sur OWL2. Nous nous sommes intéressés en particulier aux problèmes de vérification de consistance et de répondre à des requêtes dans les PDMS. Notamment, nous avons montré que ces problèmes peuvent être réduits à des raisonnements en logique propositionnelle, ce qui permet de déployer les PDMS DL-LITE$_\mathcal{R}$ au dessus des P2PIS propositionnels utilisant l'algorithme DECA$_{LR}$ pour raisonner.

Une perspective directe de ce travail serait d'étendre notre approche à des logiques de description plus expressives de la famille DL-LITE. Une autre perspective importante serait d'étudier les problèmes d'extension (non) conservative dans les PDMS DL-LITE$_\mathcal{R}$. En plus de son intérêt évoqué dans le cas des P2PIS propositionnels, la notion d'extension (non) conservative d'un pair pourrait être utile à la découverte de mappings entre ontologies (par exemple [60, 51, 58, 65, 36]) dans les PDMS [37, 30, 37, 47, 28]. En effet, il pourrait être intéressant d'exhiber les témoins de non conservativité d'un pair résultant de mappings trouvés automatiquement. En faisant cela, un expert/utilisteur pourrait choisir – de façon éclairée – les mappings acceptables en fonction de la non conservativité induite.

Enfin, la perspective de cette thèse à plus court terme est la mise en œuvre de nos algorithmes DECA$_{LR}$ et CECA dans la plateforme SOMEWHERE permettant de déployer des P2PISs propositionnels. Cette plateforme met actuellement en œuvre l'algorithme de déduction DECA passant à l'échelle d'un millier de pairs [7, 8]. Ceci permettrait d'étudier le passage à l'échelle de DECA$_{LR}$ et de CECA, ainsi que de quantifier expérimentalement le gain apporté par la déduction linéaire/DECA$_{LR}$ dans le calcul d'impliqués premiers propres.
De plus, la mise en œuvre de DECA$_{LR}$ dans SOMEWHERE permettrait d'implanter les algorithmes de gestion de données des PDMS DL-LITE$_\mathcal{R}$ comme une sur-couche logicielle des pairs, comme cela a été fait pour les PDMS SOMEOWL [8] et SOMERDFS [9]. Nous pourrions alors déployer aisément des PDMS DL-LITE$_\mathcal{R}$.

Bibliographie

[1] N. Abdallah and F.Goasdoue. Non-conservative extension of a peer in a p2p inference system. *AI communications*, 22 :211–233, 2009.

[2] N. Abdallah and F. Goasdoué. Calcul de conséquences pour le test d'extension conservatrice dans un système pair-à-pair. In *Journées francophones sur la programmation par contraintes (JFPC)*, 2008.

[3] N. Abdallah and F. Goasdoué. Systèmes pair-à-pair sémantiques et extension non conservative d'une base de connaissances. In *Base de données avancées (BDA)*, 2008.

[4] N. Abdallah, F. Goasdoué, and M.-C. Rousset. DL-LITE$_\mathcal{R}$ in the light of propositional logic for decentralized data management. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[5] N. Abdallah, F. Goasdoué, and M.-C. Rousset. Gestion décentralisée de données en DL-LITE. In *Reconnaissance des Formes et Intelligence Artificielle (RFIA)*, 2010.

[6] Andrea Acciarri, Diego Calvanese, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. Quonto : Querying ontologies. In *In Proc. of AAAI 2005*, pages 1670–1671, 2005.

[7] 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)*, pages 351–356, 2005.

[8] 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 :269–314, 2006.

[9] P. Adjiman, F. Goasdoué, and M.-C. Rousset. Somerdfs in the semantic web. *Journal on Data Semantics (JODS)*, 8 :158–181, 2007.

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

[11] E. Amir and S. McIlraith. Theorem proving with structured theories. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 624–634, 2001.

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

[13] G. Antoniou and A. Kehagias. A note on the refinement of ontologies. *International Journal of Intelligent Systems (JIS)*, 15 :623–632, 2000.

[14] V. Armant, P. Dague, and L. Simon. Distributed consistency-based diagnosis. In *International Conferences on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, pages 113–127, 2008.

[15] V. Armant, P. Dague, and L. Simon. Distributed consistency-based diagnosis without conflicts. In *International Workshop on Principles of Diagnosis (DX)*, 2008.

[16] 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.

[17] 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)*, pages 107–122, 2007.

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

[19] M. Bienvenu. Prime implicates and prime implicants in modal logic. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 379–384, 2007.

[20] Meghyn Bienvenu. Prime implicates and prime implicants : From propositional to modal logic. *Journal of Artificial Intelligence Research*, 36 :71–128, 2009.

[21] 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 (PHAI)*, pages 15–22, 2007.

[22] A. Binas and S. McIlraith. Peer-to-peer query answering with inconsistent knowledge. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 329–339, 2008.

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

[24] 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)*, pages 90–105, 2005.

[25] 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.

[26] 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) :360–384, 2008.

[27] 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)*, pages 241–251, 2004.

[28] 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)*, pages 1559–1567, 2008.

[29] L. Caroprese, C. Molinaro, and E. Zumpano. Integrating and querying p2p deductive databases. In *International Database Engineering and Applications Symposium (IDEAS)*, pages 285–290, 2006.

[30] 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)*, pages 231–250, 2003.

[31] 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)*, pages 352–356, 2006.

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

[33] S. Coste-Marquis, D. Le Berre, F. Letombe, and P. Marquis. Propositional fragments for knowledge compilation and quantified boolean formulae. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 288–293, 2005.

[34] 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.

[35] T. Dimitrakos and T. S. E. Maibaum. Notes on refinement, interpolation and uniformity. In *International Conference on Automated Software Engineering (ASE)*, page 108, 1997.

[36] A. Doan and A. Halevy. Semantic integration research in the database community : A brief survey. *AI Magazine*, 26(1) :83–94, 2005.

[37] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Y. Halevy. Learning to match ontologies on the semantic web. *VLDB Journal*, 12(4) :303–319, 2003.

[38] H. Fargier and P. Marquis. Extending the knowledge compilation map : Krom, horn, affine and beyond. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 442–447, 2008.

[39] 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)*, pages 1277–1280, 2004.

[40] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[41] 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)*, pages 187–197, 2006.

[42] 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)*, pages 717–726, 2007.

[43] 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)*, pages 298–303, 2007.

[44] 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)*, pages 198–209, 2006.

[45] A. Y. Halevy. Answering queries using views : A survey. *VLDB Journal*, 10(4) :270–294, 2001.

[46] 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)*, pages 505–517, 2003.

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

[48] J. Hladik and J. Model. Tableau systems for shio and shiq. In *International Workshop on Description Logics (DL)*, pages 168–177, 2004.

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

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

[51] Y. Kalfoglou and M. Schorlemmer. Ontology mapping : the state of the art. *The Knowledge Engineering Review*, 18(1) :1–31, 2003.

[52] R. Kontchakov, F. Wolter, and M. Zakharyaschev. Modularity in DL-lite. In *International Workshop on Description Logics (DL)*, pages 76–87, 2007.

[53] C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 453–458, 2007.

[54] C. Lutz and F. Wolter. Conservative extensions in the lightweight description logic EL. In *Conference on Automated Deduction (CADE)*, pages 84–99, 2007.

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

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

[57] 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)*, pages 604–615, 2002.

[58] N. F. Noy. Semantic integration : A survey of ontology-based approaches. *SIGMOD Record*, 33(4) :65–70, 2004.

[59] R. Pottinger and A. Y. Halevy. Minicon : A scalable algorithm for answering queries using views. *VLDB Journal*, 10(2-3) :182–198, 2001.

[60] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4) :334–350, 2001.

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

[62] M.-C. Rousset. Small can be beautiful in the semantic web. In *International Semantic Web Conference (ISWC)*, pages 6–16, 2004.

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

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

[65] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics (JODS)*, 4 :146–171, 2005.

[66] 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) :47–52, 2003.

[67] G. Tummarello, C. Morbidoni, and M. Nucci. Enabling semantic web communities with dbin : An overview. In *International Semantic Web Conference (ISWC)*, pages 943–950, 2006.