Gestion décentralisée de données en DL-LITE

Nada Abdallah[1] François Goasdoué[1] Marie-Christine Rousset[2]

[1] LRI: Univ. Paris-Sud, CNRS & INRIA
Bâtiment 490, Université Paris-Sud, 91405 Orsay Cedex
[2] LIG: Univ. Grenoble, CNRS & INRIA
681 rue de la passerelle, BP 72, 38402 St Martin d'Heres Cedex

Résumé

Cet article propose un modèle décentralisé de données et les algorithmes associés afin de mettre en œuvre des systèmes pair-à-pair de gestion de données (PDMS) fondés sur la logique de description DL-LITE$_\mathcal{R}$. *Cette logique est un fragment – ayant de bonnes propriétés théoriques et pratiques – de la prochaine recommandation du W3C pour le Web Sémantique : OWL2. Notre approche consiste à réduire la reformulation de requêtes et le test de consistance des données par rapport à une ontologie à un raisonnement en logique propositionnelle. Ceci permet de déployer de manière simple des PDMS pour* DL-LITE$_\mathcal{R}$ *"au-dessus" de SomeWhere, un système pair-à-pair d'inférence pour la logique propositionnelle passant à l'échelle du millier de pairs. Nous montrons aussi comment répondre à des requêtes à l'aide de vues – des requêtes prédéfinies – dans* DL-LITE$_\mathcal{R}$ *pour les cas centralisés et décentralisés, en combinant l'algorithme de reformulation de* DL-LITE$_\mathcal{R}$ *et l'algorithme MiniCon.*

Mots Clef

Logiques de description, logique propositionnelle, intégration d'information, systèmes pair-à-pair, web sémantique.

Abstract

This paper provides a decentralized data model and associated algorithms for peer data management systems (PDMSs) based on the DL-LITE$_\mathcal{R}$ *description logic. That logic is a fragment of the forthcoming W3C recommendation for the Semantic Web : OWL2. Our approach relies on reducing query reformulation and consistency checking for* DL-LITE$_\mathcal{R}$ *into reasoning in propositional logic. This enables a straightforward deployment of* DL-LITE$_\mathcal{R}$ *PDMSs on top of SomeWhere, a scalable peer-to-peer inference system for the propositional logic. We also show how to answer queries using views – predefined queries – in* DL-LITE$_\mathcal{R}$ *in the centralized and decentralized cases, by combining the query reformulation algorithm of* DL-LITE$_\mathcal{R}$ *and the MiniCon algorithm.*

Keywords

Description logics, propositional logic, information integration, peer-to-peer systems, semantic web.

1 Introduction

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). Celles-ci couvrent un large spectre de langages pour lesquels raisonner est 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 (*ie* 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 [4] a été spécialement définie pour garantir que répondre à une requête est polynomial dans la taille des données interrogées. 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 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 (*ie* de la Abox). Un résultat majeur de [4] est que DL-LITE$_\mathcal{R}$ est un des fragments maximaux de la famille DL-LITE pour lequel répondre à des requêtes sur des larges volumes de données est *traitable*. DL-LITE$_\mathcal{R}$ est un fragment du langage d'ontologies OWL[1] du W3C et de son évolution à venir OWL2[2]. De plus, DL-

[1] http ://www.w3.org/2004/OWL/
[2] http ://www.w3.org/TR/owl2-overview/

LITE$_\mathcal{R}$ étend RDFS[3], avec la possibilité de déclarer des relations disjointes ou d'exprimer des restrictions de cardinalités. RDFS est le standard du W3C pour décrire les méta-données des ressources du Web Sémantique.

Pour des raisons de passage à l'échelle, de robustesse et de protection de données, il est 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. Chaque pair peut avoir sa propre ontologie pour décrire ses données et interagit avec d'autres pairs en établissant des *mappings* – connexions sémantiques – 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.

La contribution de cet article est un modèle de données décentralisé et les algorithmes associés de gestion de données dans un Web sémantique fondé sur la logique DL-LITE$_\mathcal{R}$ *distribuée*. Nous revisitons l'approche centralisée de [4], pour la consistance des données et la réponse à des requêtes par reformulation, dans le but de définir les algorithmes décentralisés correspondant. Nous étendons aussi les travaux sur DL-LITE en fournissant un algorithme centralisé et un algorithme décentralisé pour répondre à des requêtes à l'aide de vues – requêtes prédéfinies –, lorsque requêtes et vues sont des requêtes conjonctives (*ie* selection-projection-jointure) sur des ontologies en DL-LITE$_\mathcal{R}$.

Notre approche consiste à réduire les problèmes de gestion de données en DL-LITE$_\mathcal{R}$ mentionnés ci-dessus en des raisonnements décentralisés en logique propositionnelle. Cela permet alors de déployer des systèmes de gestion de données de DL-LITE$_\mathcal{R}$ au-dessus de la plate-forme SomeWhere, un système pair-à-pair d'inférence dont des expériences ont démontré le passage à l'échelle du millier de pairs [1].

L'article est organisé comme suit. Dans la Section 2, nous présentons le modèle de données distribué que nous proposons pour DL-LITE$_\mathcal{R}$. Ensuite, dans la Section 3, nous fournissons des algorithmes décentralisés pour le test de consistance et pour répondre à une requête par reformulation. Nous étudions le problème de répondre à une requête en termes de vues dans les cas centralisé et décentralisé dans la Section 4. Enfin, nous concluons brièvement dans la Section 5.

2 Modèle DL-LITE$_\mathcal{R}$ distribué

La majorité des LD 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* (*ie* un concept atomique A ou une *quantification existentielle non qualifiée sur un rôle basique* $\exists R$) et R un *rôle basique* (*ie* un rôle atomique P ou son inverse P^-). Enfin, C est un *concept général* (*ie* un concept basique ou sa négation) et E un *rôle général* (*ie* 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 et une relation binaire sur Δ^I à chaque rôle atomique. La sémantique des concepts et rôles non atomiques est définie comme suit :
$(P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\}$
$(\exists R)^I = \{o_1 \mid \exists o_2\ (o_1, o_2) \in R^I\}$
$(\neg B)^I = \Delta^I \backslash B^I$ and $(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$
Une interprétation I est un *modèle d'un concept* C *(resp. un rôle* E*)* si $C^I \neq \emptyset$ (resp. $E^I \neq \emptyset$).

Bases de connaissances. Une base de connaissances DL-LITE$_\mathcal{R}$ (BC) est constituée d'une *Tbox* représentant une vue conceptuelle d'un domaine d'intérêt (*ie* une ontologie) et soit d'une *Abox* (un ensembles de faits) [4], soit d'*extensions de vues* (un ensemble de réponses à des requêtes prédéfinies) [6], pour représenter les données.

Une Tbox 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 appelées 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 negatives (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 $I = (\Delta^I, .^I)$ est un *modèle d'une inclusion* $B \sqsubseteq C$ (resp. $R \sqsubseteq E$) si $B^I \subseteq C^I$ (resp. $R^I \subseteq E^I$). C'est un *modèle d'une Tbox* si I satisfait toutes les inclusions. 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 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 $a^I \in \Delta^I$ (*ie* l'hypothèse du nom unique est vérifiée). Une interprétation I 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.

Lorsque les données sont modélisées à l'aide d'extensions de vues, la BC est de la forme $\langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ telle que $\mathcal{E}$ est un ensemble de faits de la forme $v(\bar{t})$ où v est une *vue* de $\mathcal{V}$.

Requêtes et vues sur une BC. Nous considérons ici des (unions de) *requêtes conjonctives*, c'est-à-dire des requêtes de type selection-projection-jointure, *ie* 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

[3] http ://www.w3.org/TR/rdf-schema/

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 ex. 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 : $q^I = \{\bar{e} \in (\Delta^I)^n \mid [\exists \bar{y}\ conj(\bar{e}, \bar{y})]^I = true\}$.
Une *vue* v est définie par une requête conjonctive $v(\bar{x}) = \exists \bar{y}\ conj(\bar{x}, \bar{y})$ et possède une extension $\mathcal{E}(v)$ qui est un ensemble de faits de la forme $v(\bar{t})$. En suivant l'*hypothèse du monde ouvert*, nous adoptons une sémantique *correcte* pour les vues, *ie* 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.
Un *modèle d'une BC* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (resp. $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$) est une interprétation I qui est à la fois un modèle de $\mathcal{T}$ et de $\mathcal{A}$ (resp. de $\mathcal{T}$, $\mathcal{V}$ et $\mathcal{E}$). Une BC $\mathcal{K}$ est *consistante* si elle a au moins un modèle. Enfin, une BC $\mathcal{K}$ *implique logiquement* un fait β, noté $\mathcal{K} \models \beta$, si tout modèle de $\mathcal{K}$ est un modèle de β.

Réponses (certaines) à une requête sur une BC. Pour définir les réponses à une requête sur une BC, il faut distinguer le cas où l'extension des prédicats utilisés dans la requête est fournie par une Abox, du cas où les données peuvent seulement être (partiellement) inférées à partir des extensions de vues. Dans ce dernier cas, on parle de *réponses certaines*.
L'ensemble de réponses à une requête n-aire non booléenne q sur $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ est : $ans(q, \mathcal{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{K} \models q(\bar{t})\}$, 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.
De façon similaire, les réponses certaines à une requête n-aire non booléenne q sur $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ est : $cert(q, \mathcal{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{K} \models q(\bar{t})\}$, où $\mathcal{C}$ est l'ensemble des constantes de $\mathcal{E}$ et $q(\bar{t})$ est la formule *close* obtenue comme ci-dessus. Par convention, l'ensemble des réponses (certaines) à une requête booléenne est $\{()\}$, où $()$ est le tuple vide, si $\mathcal{K} \models q()$, et $\emptyset$ sinon.

PDMS. Un PDMS $\mathcal{S}$ est un ensemble de pairs $\{\mathcal{P}_i\}_{i=1..n}$, où l'indice i modélise l'identifiant du pair $\mathcal{P}_i$ (par ex. son adresse IP). Chaque pair $\mathcal{P}_i$ gère sa propre BC $\mathcal{K}_i$ exprimée en termes de son propre *vocabulaire*, *ie* ses concepts et rôles atomiques. Nous notons A_i (resp. P_i) le concept atomique A (resp. le rôle atomique P) de $\mathcal{P}_i$. De façon similaire, nous notons v_i la vue v 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) $\mathcal{K} = \bigcup_{i=1}^{n} \mathcal{K}_i$. C'est-à-dire que, contrairement à d'autres approches (par ex. [5]), nous adoptons une sémantique logique standard pour les mappings.

Exemple La figure 1 représente le PDMS $\mathcal{S}$ composé des pairs $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ et $\mathcal{P}_4$. 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 KB respective. Nous utilisons les lettres majuscules pour les concepts et les lettres miniscules pour les rôles.

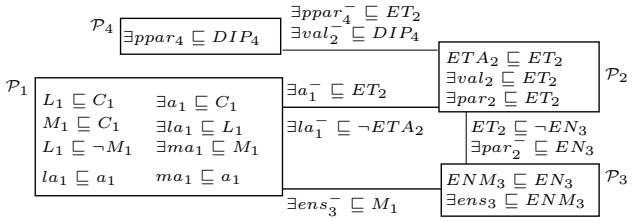

FIG. 1 – Le PDMS $\mathcal{S}$

Le PDMS modélise une application décentralisée de services de cours particuliers. $\mathcal{P}_1$ gère les cours (C_1) comprenant des cours de langues ($L_1 \sqsubseteq C_1$) et de mathématiques ($M_1 \sqsubseteq C_1$) qui sont disjoints ($L_1 \sqsubseteq \neg M_1$), $\mathcal{P}_2$ gère les étudiants (ET_2) dont certains sont anglophones ($ETA_2 \sqsubseteq ET_2$), $\mathcal{P}_3$ gère les enseignants (EN_3) dont une partie sont des mathématiciens ($ENM_3 \sqsubseteq EN_3$) et $\mathcal{P}_4$ gère les diplômes (DIP_4).
Des cours de $\mathcal{P}_1$ sont offerts à des étudiants de $\mathcal{P}_2$ ainsi le rôle a_1 est défini entre $\mathcal{P}_1$ et $\mathcal{P}_2$ ($\exists a_1 \sqsubseteq C_1$ et $\exists a_1^- \sqsubseteq ET_2$). Toutefois, des cours de langues ne sont pas offerts à des anglophones ($la_1 \sqsubseteq a_1$, $\exists la_1 \sqsubseteq L_1$, et $\exists la_1^- \sqsubseteq \neg ETA_2$). Les étudiants de $\mathcal{P}_2$ et les enseignants de $\mathcal{P}_3$ sont disjoints ($ET_2 \sqsubseteq \neg EN_3$). De plus, des étudiants de $\mathcal{P}_2$ sont enseignés par des enseignants de $\mathcal{P}_3$ ($\exists par_2 \sqsubseteq ET_2$ et $\exists par_2^- \sqsubseteq EN_3$). Des mathématiciens de $\mathcal{P}_3$ enseignent des cours de math de $\mathcal{P}_1$ ($\exists ens_3 \sqsubseteq ENM_3$ et $\exists ens_3^- \sqsubseteq M_1$). Enfin, des diplômes de $\mathcal{P}_4$ sont préparés par des étudiants de $\mathcal{P}_2$ ($\exists ppar_4 \sqsubseteq DIP_4$ et $\exists ppar_4^- \sqsubseteq ET_2$) et des étudiants de $\mathcal{P}_2$ valident des diplômes de $\mathcal{P}_4$ ($\exists val_2 \sqsubseteq ET_2$ et $\exists val_2^- \sqsubseteq DIP_4$).

3 Répondre à des requêtes

Nous commençons par rappeler les algorithmes centralisés $Answer$, $Consistent$ et $PerfectRef$ de [4] qui sont utilisés pour répondre à des requêtes sur une BC $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (Section 3.1). Nous en fournissons ensuite les versions décentralisées (dans les Sections 3.3 et 3.4). Elles sont basées sur un encodage propositionnel présenté en Section 3.2 et utilisent l'algorithme décentralisé DeCA [1] de raisonnement en logique propositionnelle.

3.1 Algorithmes centralisés pour DL-LITE$_\mathcal{R}$

Étant donné une union de requêtes conjonctives Q sur une BC $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $Answer$ (Algorithme 1) vérifie d'abord si $\mathcal{K}$ est inconsistante (ligne 1). Dans ce cas, il retourne

tous les tuples de l'arité de Q pouvant être générés à partir des constantes apparaissant dans $\mathcal{A}$ (ligne 2). Sinon, il obtient $ans(Q, \mathcal{K})$ en évaluant sur $\mathcal{A}$ – considérée comme une base de données relationnelles – l'union de requêtes conjonctives obtenue par reformulation de Q (ligne 3).

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

$Consistent$ (Algorithme 2) 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 en logique du premier ordre* (lignes 1 à 3) de la clôture des NI de $\mathcal{T}$, notée $cln(\mathcal{T})$, *ie* l'ensemble de toutes les NI logiquement impliquées par $\mathcal{T}$. Cette traduction des NI 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 $B_i = \exists P_i$
$\gamma_i(x) = \exists y_i\ P_i(y_i, x)$ 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 $R_i = P_i$
$\rho_i(x, y) = P_i(y, x)$ si $R_i = P_i^-$

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

Enfin, $PerfectRef$ (Algorithme 3) reformule chaque requête conjonctive q de Q 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 (*ie* il n'y a pas de jointure sur cette variable).
La définition suivante (Definition 32 dans [4]) définit le résultat $gr(g, I)$ de l'application de la PI I à l'atome g. Cette opération est centrale dans $PerfectRef$ pour la reformulation d'une requête (boucle (a), lignes 5 à 7).

Définition 1 (Application d'une PI à un atome) *Soit I une inclusion applicable à un atome g. $gr(g, I)$ est l'atome défini comme suit :*
- si $g = A(x)$ et $I = A_1 \sqsubseteq A$ alors $gr(g, I) = A_1(x)$
- si $g = A(x)$ et $I = \exists P \sqsubseteq A$ alors $gr(g, I) = P(x, \_)$
- si $g = A(x)$ et $I = \exists P^- \sqsubseteq A$ alors $gr(g, I) = P(\_, x)$
- si $g = P(x, \_)$ et $I = A \sqsubseteq \exists P$ alors $gr(g, I) = A(x)$
- si $g = P(x, \_)$ et $I = \exists P_1 \sqsubseteq \exists P$ alors $gr(g, I) = P_1(x, \_)$
- si $g = P(x, \_)$ et $I = \exists P_1^- \sqsubseteq \exists P$ alors $gr(g, I) = P_1(\_, x)$
- si $g = P(\_, x)$ et $I = A \sqsubseteq \exists P^-$ alors $gr(g, I) = A(x)$
- si $g = P(\_, x)$ et $I = \exists P_1 \sqsubseteq \exists P^-$ alors $gr(g, I) = P_1(x, \_)$
- si $g = P(\_, x)$ et $I = \exists P_1^- \sqsubseteq \exists P^-$ alors $gr(g, I) = P_1(\_, x)$
- si $g = P(x_1, x_2)$ et soit $I = P_1 \sqsubseteq P$ ou $I = P_1^- \sqsubseteq P^-$ alors $gr(g, I) = P_1(x_1, x_2)$
- si $g = P(x_1, x_2)$ et soit $I = P_1 \sqsubseteq P^-$ ou $I = P_1^- \sqsubseteq P$ alors $gr(g, I) = P_1(x_2, x_1)$

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 3: 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** $PR' = PR$

Exemple (suite). Considérons le PDMS de la figure 1 comme une BC centralisée $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ avec $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 \cup \mathcal{T}_4$ et $\mathcal{A} = \{a_1(Algebre, Marc), par_2(Pierre, Vincent), ens_3(Marc, Analyse), ppar_4(CS, Pierre)\}$.
Étant donnée la requête $q(x) = DIP_4(x)$, $Answer(q, \mathcal{K})$ teste tout d'abord si $\mathcal{K}$ est consistante. $Consistent(\mathcal{K})$ calcule dans un premier temps $cln(\mathcal{T})$, qui contient $\exists a_1^- \sqsubseteq \neg\exists ens_3 \in cln(\mathcal{T})$ – à cause de $\exists a_1^- \sqsubseteq ET_2$, $ET_2 \sqsubseteq \neg EN_3$, $ENM_3 \sqsubseteq EN_3$, et $\exists ens_3 \sqsubseteq ENM_3$ – et dont la traduction en logique du premier ordre est $\exists x(\exists y_1 a_1(y_1, x) \wedge \exists y_2 ens_3(x, y_2))$. Par conséquent, $q_{unsat}^{db(\mathcal{A})}$ s'évalue à $true$ et $Consistent(\mathcal{K})$ retourne $false$. $Answer(q, \mathcal{K})$ retourne donc $Alltup(q, \mathcal{K})$.

Si nous considérons maintenant la Abox $\mathcal{A} = \{a_1(Algebre, Marc), par_2(Pierre, Vincent), ens_3(Vincent, Analyse)\}$ alors $\mathcal{K}$ est consistante et $Answer(q, \mathcal{K})$ calcule la reformulation $\{DIP_4(x)\} \cup \{ppar_4(x, \_)\} \cup \{val_2(\_, x)\}$ puisque $\exists ppar_4 \sqsubseteq DIP_4$ et $\exists val_2^- \sqsubseteq DIP_4$ sont applicables à $DIP_4(x)$. $Answer(q, \mathcal{K})$ retourne donc $\{(CS)\}$.

3.2 Encodage propositionnel d'une Tbox

L'encodage d'une Tbox, noté $\Phi(\mathcal{T})$, est la théorie de logique propotitionnelle 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 :

$\Phi(P \sqsubseteq Q)$=$\{P \Rightarrow Q, P^- \Rightarrow Q^-, P^{\exists} \Rightarrow Q^{\exists}, P^{\exists^-} \Rightarrow Q^{\exists^-}\}$
$\Phi(P^- \sqsubseteq Q)$=$\{P^- \Rightarrow Q, P \Rightarrow Q^-, P^{\exists^-} \Rightarrow Q^{\exists}, P^{\exists} \Rightarrow Q^{\exists^-}\}$
$\Phi(P \sqsubseteq Q^-)$=$\{P \Rightarrow Q^-, P^- \Rightarrow Q, P^{\exists} \Rightarrow Q^{\exists^-}, P^{\exists^-} \Rightarrow Q^{\exists}\}$
$\Phi(P^- \sqsubseteq Q^-)$=$\{P^- \Rightarrow Q^-, P \Rightarrow Q, P^{\exists} \Rightarrow Q^{\exists}, P^{\exists^-} \Rightarrow Q^{\exists^-}\}$
$\Phi(P \sqsubseteq \neg Q)$=$\{P \Rightarrow \neg Q, P^- \Rightarrow \neg Q^-\}$
$\Phi(P^- \sqsubseteq \neg Q)$=$\{P^- \Rightarrow \neg Q, P \Rightarrow \neg Q^-\}$
$\Phi(P \sqsubseteq \neg Q^-)$=$\{P \Rightarrow \neg Q^-, P^- \Rightarrow \neg Q\}$
$\Phi(P^- \sqsubseteq \neg Q^-)$=$\{P^- \Rightarrow \neg Q^-, P \Rightarrow \neg Q\}$

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 $R = P^-$.

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

Dans la suite, nous utiliserons l'algorithme DECA pour raisonner sur l'encodage d'une Tbox distribuée. Cet algorithme décentralisé est implémenté dans la plate-forme SomeWhere [1] : un système pair-à-pair d'inférence qui calcule d'une façon décentralisée des conséquences logiques d'une théorie clausale propositionnelle distribuée dans le système. Plus précisément, grâce à une copie de DECA s'exécutant localement sur chaque pair, $DeCA^i(l_i)$ (denotant DECA s'exécutant sur le pair $\mathcal{P}_i$ avec comme entrée le littéral l_i du vocabulaire de $\mathcal{P}_i$) produit l'ensemble de tous les *impliqués premiers propres* de l_i par rapport à la théorie distribuée $\bigcup_{i=1}^n \Phi(\mathcal{T}_i)$, *ie* les plus fortes conséquences logiques de $\{l_i\} \cup \bigcup_{i=1}^n \Phi(\mathcal{T}_i)$, qui ne sont pas conséquences de la seule théorie $\bigcup_{i=1}^n \Phi(\mathcal{T}_i)$.

3.3 Vérification décentralisée de consistance

Notre approche consiste à décentraliser le calcul de la clôture des NI de la Tbox distribuée $\bigcup_{i=1}^n \mathcal{T}_i$ d'un PDMS sans rôle vide[4] (*ie* où rien n'interdit d'avoir des instances pour les rôles) en exploitant la propriété de tranfert de l'encodage propositionnel (Théorème 1) et DECA. Le point subtil est que dans un cadre décentralisé, nous devons déclencher le calcul de la clôture des NI sur chaque pair 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 2) et calculons avec DECA la *clôture des NI d'un pair* par rapport à un PDMS sans rôle vide.

Théorème 1 (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 distincts ou deux rôles basiques distincts : $cln(\mathcal{T}) \models X \sqsubseteq \neg Y$ ssi $\Phi(\mathcal{T}) \models \neg\Phi(X) \vee \neg\Phi(Y)$.*

La preuve s'effectue par récurrence sur le nombre de règles définissant la clôture des NI (Définition 9 dans [4]) utilisées pour produire $X \sqsubseteq \neg Y$ ($\Rightarrow$) et sur la plus petite taille de preuve par résolution pour produire $\neg\Phi(X) \vee \neg\Phi(Y)$ ($\Leftarrow$).

Définition 2 (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 *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 $\neg\Phi(X) \in DeCA^j(\Phi(Y))$.*
- *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 $\neg\Phi(X) \in DeCA^j(\neg\Phi(Y))$*
  - *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 $\neg\Phi(X) \in DeCA^j(\neg\Phi(Z))$.*

La version décentralisée de l'algorithme $Consistent$ original, notée $Consistent^i$ lorsqu'elle s'exécute sur le pair $\mathcal{P}_i$, est obtenue en remplaçant **foreach** $\alpha \in cln(\mathcal{T})$ à la ligne 2 de **Algorithme 2** 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.

Le théorème 2 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.

Théorème 2 (Correction de la vérification de 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 $i = 1..n$.*

[4] Pour des contraintes d'espace, nous ne développons pas les raisons de cette contrainte.

La preuve repose tout d'abord sur le Théorème 1 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, le Lemme 12 de [4] et la complétude de DECA (prouvée dans [1]) garantissent que $cln(\mathcal{P}_i)$ définie dans la Définition 2 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. Par conséquent, le Théorème 15 et le Lemme 16 de [4] garantissent 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.

3.4 Reformulation decentralisée de requêtes

Notre approche s'appuie sur l'encodage propositionnel et l'utilisation de DECA afin de décentraliser la *clôture en chaînage arrière par rapport à des PI* de chacun des atomes de la requête. La Définition 3 définit la *clôture en chaînage arrière* d'un atome par rapport à des PI comme l'itération d'étapes élémentaires d'application de PI en chaînage arrière (Définition 1). La Proposition 1 établit la terminaison de ce processus itératif.

Définition 3 (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 réursivement comme suit :*

- $cl\_gr^1(\mathcal{A}, PI) = \{gr(g, I) \mid g \in \mathcal{A},\ I \in PI$ *et I est applicable à* $g\}$
- $cl\_gr^{i+1}(\mathcal{A}, PI) = cl\_gr^1(cl\_gr^i(\mathcal{A}, PI), PI)$.

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

La preuve correspond à la preuve de terminaison de $PerfectRef$ (Lemme 34 dans [4]).
Le Théorème 3 est l'équivalent pour les PI de la propriété de transfert de l'encodage propositionnel pour les NI (Théorème 1). Sa preuve est aussi par récurrence (nombre d'applications de PI et plus petites preuves par résolution).

Théorème 3 (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 :*
- $g = A(x)$, $g' = A'(x)$, $V_1 = A$ *et* $V_2 = A'$;
- $g = A(x)$, $g' = P(x, \_)$, $V_1 = A$ *et* $V_2 = P^{\exists}$;
- $g = A(x)$, $g' = P(\_, x)$, $V_1 = A$ *et* $V_2 = P^{\exists^-}$;
- $g = P(x, y)$, $g' = Q(x, y)$, $V_1 = P$ *et* $V_2 = Q$;
- $g = P(x, y)$, $g' = Q(y, x)$, $V_1 = P$ *et* $V_2 = Q^-$;
- $g = P(x, \_)$, $g' = A(x)$, $V_1 = P^{\exists}$ *et* $V_2 = A$;
- $g = P(x, \_)$, $g' = Q(x, \_)$, $V_1 = P^{\exists}$ *et* $V_2 = Q^{\exists}$;
- $g = P(x, \_)$, $g' = Q(\_, x)$, $V_1 = P^{\exists}$ *et* $V_2 = Q^{\exists^-}$;
- $g = P(\_, x)$, $g' = A(x)$, $V_1 = P^{\exists^-}$ *et* $V_2 = A$;
- $g = P(\_, x)$, $g' = Q(x, \_)$, $V_1 = P^{\exists^-}$ *et* $V_2 = Q^{\exists}$;
- $g = P(\_, x)$, $g' = Q(\_, x)$, $V_1 = P^{\exists^-}$ *et* $V_2 = Q^{\exists^-}$.

Étant donné le Théorème 3, le calcul décentralisé de $cl\_gr(g, PI)$ est évident en utilisant DECA : si g est construit en termes du vocabulaire du pair $\mathcal{P}_i$, $g' \in cl\_gr(g, PI)$ ssi $\neg V_2 \in DeCA^i(\neg V_1)$ pour les mêmes quadruplets de valeurs de g, g', V_1 et V_2 que dans le Théorème 3.
La version décentralisée de $PerfectRef$, notée $PerfectRef^i$ lorsqu'elle s'exécute sur le pair $\mathcal{P}_i$, est fournie par l'Algorithme 4. 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 4: 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) $PR'' := \oslash_{i=1}^{n} cl\_gr(g_i, PI)$
(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 $PerfectRef^i$.

Théorème 4 (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 3 et de la complétude de DECA, garantissant le calcul décentralisé de tout l'ensemble $cl\_gr(g_i, PI)$.

3.5 Répondre à des requêtes de façon décentralisée

Contrairement à l'algorithme $Answer$ original, la consistance globale du PDMS ne peut pas être vérifiée car le pair interrogé $\mathcal{P}_i$ ne connaît pas tous les pairs du PDMS. Toutefois, il peut obtenir les identifiants $1, \ldots, k$ des pairs 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. L'algorithme 5 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 à la requête en évaluant l'union de ses reformulations sur les Aboxes pertinentes, *ie* $\bigcup_{j=[1..k]} \mathcal{A}_j$. Dans cet algorithme $\perp$ remplace $AllTup(Q, \mathcal{K})$ par rapport à l'algorithme $Answer$ original. L'intérêt de l'algorithme 5 est de fournir des réponses *bien fondées*, *ie* des réponses dérivées à partir d'un sous-ensemble consistant de la KB (éventuellement inconsistante) du PDMS.

Algorithme 5: 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 $\mathcal{K}_i = \langle \mathcal{T}_i, \mathcal{A}_i \rangle$ de $\mathcal{P}_i$

Sortie: $ans(Q, \mathcal{K})$ où $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ est la BC du PDMS $\mathcal{S}$

(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** $q^{db(\bigcup_{j=1}^{k} \mathcal{A}_j)}$ tq $1, \ldots, k$ sont tous les identifiants de pairs apparaissant dans q

(4) **else return** le singleton $\{\perp\}$

Exemple (suite). Considérons le PDMS de la figure 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 $Consistent$, $Answer^2(q)$ sait que $\mathcal{P}_1$ contribue à l'inconsistance du système ($Consistent^1$ trouve $\exists a_1^- \sqsubseteq \neg\exists ens_3 \in cln(\mathcal{P}_1)$). $Answer^2(q, \mathcal{K})$ 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 $\{(CS)\}$ qui est une réponse bien fondée.

4 Répondre à des requêtes via des vues

Nous fournissons des algorithmes pour calculer les réponses certaines à une requête sur une BC (centralisée ou distribuée) $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. Pour cela, nous utilisons l'algorithme $MiniCon$ [8] qui produit les réécritures maximales d'une requête conjonctive q en utilisant l'ensemble $\mathcal{V}$ de vues conjonctives. 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 $\mathcal{T} \cup \mathcal{V} \models \forall \bar{x}\,(q_v(\bar{x}) \Rightarrow q(\bar{x}))$. Une réécriture q_v de q est *maximale* s'il n'existe pas d'autre réécriture de q strictement plus générale que q_v. Il a été montré ([7]) 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.

4.1 Cas centralisé

$ViewConsistent$ est une variante de l'algorithme $Consistent$ original permettant de vérifier la consistance d'une BC $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. Il est obtenu en remplaçant :
- $q_{unsat} := q_{unsat} \cup \{\delta(\alpha)\}$ à la ligne 3 de l'Algorithme 2 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 2 par $q_{unsat}^{db(\mathcal{E})}$, *ie* l'évaluation de q_{unsat} – au sens des bases de données relationnelles – sur les extensions de vues.

Théorème 5 (Correction de la vérification de consistance) *Soit $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ une BC. $\mathcal{K}$ est consistante ssi $ViewConsistent(\mathcal{K})$ retourne $true$.*

Considérons de façon équivalente que $\mathcal{K}$ est inconsistante ssi $ViewConsistent(\mathcal{K})$ retourne $false$.
($\Leftarrow$) 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 $\mathcal{K}$ est inconsistante dès lors que $ViewConsistent(\mathcal{K})$ retourne $false$.
($\Rightarrow$) Observons que $\mathcal{T} \cup \mathcal{V}$ et $\mathcal{V} \cup \mathcal{E}$ sont toujours satisfiables. Donc, si $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ 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 adaptant la notion de témoin d'un tuple introduite dans le Lemme 39 de [4] à $\mathcal{E}$ 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 [8]), q_v spécialise l'une d'elles, donc $ViewConsistent(\mathcal{K})$ retourne $false$.

$CertainAnswer$ (Algorithme 6) est une variante de l'algorithme $Answer$ original qui calcule les réponses cer-

taines d'une union de requêtes conjonctives Q sur une BC $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$. 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 $PerfectRef$, modélisant les différentes manières de déplier la requête initiale à partir des PI de $\mathcal{T}$.

Algorithme 6: L'algorithme $CertainAnswer$
$CertainAnswer(Q, \mathcal{K})$
Entrée: une union de requêtes conjonctives Q et une BC $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$
Sortie: $cert(Q, \mathcal{K})$
(1) **if** not $ViewConsistent(\mathcal{K})$
(2) **return** $Alltup(Q, \mathcal{E})$
(3) **else**
(4) $Q' := \bigcup_{q \in Q} PerfectRef(q, \mathcal{T})$
(5) **return** $(\bigcup_{q' \in Q'} MiniCon(q', \mathcal{V}))^{db(\mathcal{E})}$

Théorème 6 (Correction de $CertainAnswer$) *Soient $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ une BC et Q une union de requêtes conjonctives. $cert(Q, \mathcal{K}) = CertainAnswer(Q, \mathcal{K})$.*

Quand $\mathcal{K}$ est inconsistante, la correction de $CertainAnswer$ découle directement de celle de $ViewConsistent$. Considérons maintenant que $\mathcal{K}$ 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 adaptant la notion de témoin d'un tuple introduite dans le Lemme 39 de [4] à $\mathcal{E}$ 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 [8]), 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 6.

4.2 Cas décentralisé

Pour des contraintes d'espace, nous ne présentons que brièvement l'extension des résultats précédents aux PDMS.
Les versions décentralisées de $ViewConsistent$ et de $CertainAnswer$ sont notées $ViewConsistent^i$ et $CertainAnswer^i$ lorsqu'elles s'exécutent sur un pair $\mathcal{P}_i$. Ce sont des variantes des algorithmes décentralisés $Consistent^i$ et $Answer^i$ présentés dans la section précédente. Elles utilisent la fonction $fetchViews(q)$ où q est une requête conjonctive pouvant mettre en jeu le vocabulaire de différents pairs : $fetchViews(q)$ récupère sur ces pairs les 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.

L'algorithme $ViewConsistent^i$ 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 : $Q_{unsat} := \bigcup_{q \in q_{unsat}} MiniCon(q, fetchViews(q))$.
L'algorithme $CertainAnswer^i$ 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 5) par l'évaluation de la requête $Q' := \bigcup_{q' \in q} MiniCon(q', fetchViews(q'))$ sur les extensions de vues pertinentes.

5 Conclusion

Cet article se fonde sur et étend des travaux en intégration d'information ([4, 6, 8]). Pour répondre à des requêtes à l'aide de vues dans DL-LITE$_\mathcal{R}$, nous fournissons un algorithme centralisé et un autre décentralisé qui calculent les réponses certaines via des réécritures. Dans le cas décentralisé, nos travaux étendent le modèle de données des PDMS SOMERDFS [2]. Enfin, notre notion de réponses bien fondées se distingue des travaux de [3] qui utilise l'*answer set programming* afin de définir des réponses consistantes dans un PDMS éventuellement inconsistant.

Références

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a p2p setting : Application to the semantic web. *Journal of Artificial Intelligence Research (JAIR)*, 25 :269–314, 2006.

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

[3] L. E. Bertossi and L. Bravo. The semantics of consistency and trust in peer data exchange systems. In *Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, pages 107–122, 2007.

[4] 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.

[5] 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.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. View-based query answering over description logic ontologies. In *Principles of Knowledge Representation and Reasoning (KR)*, 2008.

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

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