

Découverte de correspondances entre ontologies distribuées

François-Élie Calvier, Chantal Reynaud

LRI, Univ. Paris-Sud & INRIA Futurs, Bât. G, 2-4 rue Jacques Monod, 91893 Orsay Cedex, France

Résumé : Dans cet article, nous nous intéressons à la découverte de mises en correspondance entre ontologies modélisant les connaissances des pairs d'un système pair à pair de gestion de données (PDMS). Ce travail est réalisé dans le cadre du projet MediaD au sein de la plate-forme SomeRDFS dans le contexte duquel les ontologies, les données et les mappings sont exprimés en RDF(S). Nous présentons ce contexte dans une première partie. Nous précisons ensuite les spécificités du processus de mise en correspondance dans le cadre de PDMS par opposition au contexte centralisé. Nous montrons enfin comment le raisonnement effectué dans un PDMS peut aider à découvrir des mappings entre ontologies et décrivons quelques techniques de découvertes de mappings utilisables.

Mots-clés : ontologies, mappings, alignement, système pair à pair de gestion de données

1 Introduction

Nous nous intéressons à la découverte de correspondances, ou mappings, entre ontologies modélisant les connaissances de pairs d'un PDMS (système pair à pair de gestion de données). Un PDMS est un système constitué de pairs autonomes qui communiquent pour répondre collectivement à une requête. Les communications entre pairs s'établissent grâce à des mappings qui définissent des relations sémantiques entre leurs connaissances. Un PDMS est sollicité via l'interrogation d'un des pairs qui pourra ensuite faire appel aux autres pour répondre. Une spécificité des PDMS est que chaque pair ne connaît que ses propres connaissances et les mappings le connectant à d'autres pairs. Nous présenterons dans un premier temps le contexte de notre travail. Nous préciserons ensuite les spécificités du processus de mise en correspondance dans le cadre des PDMS par opposition au contexte centralisé. Nous montrerons enfin comment les réponses aux requêtes peuvent être exploitées pour identifier les éléments pour lesquels la recherche de mappings est pertinente, et quelles techniques mettre en place pour découvrir des mappings nouveaux.

2 Contexte de travail

Nous travaillons, dans le cadre du projet MediaD¹, dont l'objectif est la création d'un environnement déclaratif de construction de systèmes de gestion de données P2P, exploitant des modèles de données exprimés en logique propositionnelle. Le but consiste à permettre le déploiement de très grosses applications faisant intervenir des milliers de pairs. Les premiers travaux ont conduit à la construction de la plateforme SomeWhere. Ces travaux se sont ensuite poursuivis montrant que RDF(S) pouvait être utilisé pour représenter des ontologies distribuées au sein de SomeWhere. Ceux-ci ont conduit au développement de la plate-forme SomeRDFS (Adjiman *et al.*, 2006) au sein de laquelle nous situons notre travail. Dans le contexte de SomeRDFS, les ontologies, les données et les mappings sont exprimés en RDF(S). Il est ainsi possible de définir des classes, des sous-classes, des propriétés, des sous-propriétés, de typer le domaine et le co-domaine² des propriétés. Les constructeurs autorisés sont l'inclusion classes, l'inclusion de propriétés, le typage de domaine et de co-domaine. Ce langage, restreint à ces constructeurs, appelé core-RDFS, a l'avantage d'avoir une sémantique logique claire et intuitive. Il est basé sur des relations unaires qui représentent les classes et des relations binaires qui représentent des propriétés. Nous donnons dans TAB. 1 la sémantique logique de RDF(S) en indiquant la notation en logique de description associée (notation LD) et la traduction en logique du 1er ordre des constructeurs correspondant.

Opérateur	Notation LD	Traduction en logique du premier ordre
Inclusion de classes	$C_1 \sqsubseteq C_2$	$\forall X, C_1(X) \Rightarrow C_2(X)$
Inclusion de propriétés	$P_1 \sqsubseteq P_2$	$\forall X \forall Y R_1(X, Y) \Rightarrow R_2(X, Y)$
Typage de domaine d'une propriété	$\exists P \sqsubseteq C$	$\forall X \forall Y, P(X, Y) \Rightarrow C(X)$
Typage de co-domaine d'une propriété	$\exists P^- \sqsubseteq C$	$\forall X \forall Y, P(X, Y) \Rightarrow C(Y)$

TAB. 1 – Opérateurs RDF(S)

Les ontologies des pairs sont représentées à l'aide d'expressions core-RDFS composées uniquement de relations appartenant au vocabulaire du pair. Le vocabulaire d'un pair comprend l'union de l'ensemble des noms de classes et de l'ensemble des noms de propriétés. Les noms de classes et de propriétés sont propres à un pair donné. Nous notons $\mathcal{P}:R$ la relation R (classe ou propriété) de l'ontologie du pair \mathcal{P} .

Les données des pairs sont associées à des relations faisant partie de son vocabulaire. Ainsi, les notations en logique terminologique et en logique du 1er ordre correspondant respectivement à la déclaration d'une instance d'une classe et d'une instance d'une propriété sont $C(a)$ et $P(a,b)$, a et b étant des constantes.

Un mapping correspond à une inclusion entre classes ou propriétés de 2 pairs différents (cf. TAB. 2 a et b) ou au typage d'une propriété d'un pair donné avec une classe d'un autre pair (cf. TAB. 2 c et d). Ainsi les mappings sont également des ex-

¹Projet financé par France Telecom R&D dans le cadre d'un CRE (2004-2008)

²Traduction de "range"

pressions RDF(S). Leur spécificité est d'être construites à partir du vocabulaire des ontologies de pairs différents qui établissent ainsi des correspondances sémantiques entre eux.

Mappings	Notation LD	Traduction en logique du premier ordre
a. Inclusion de classes	$\mathcal{P}_1:C_1 \sqsubseteq \mathcal{P}_2:C_2$	$\forall X, \mathcal{P}_1:C_1(X) \Rightarrow \mathcal{P}_2:C_2(X)$
b. Inclusion de propriétés	$\mathcal{P}_1:P_1 \sqsubseteq \mathcal{P}_2:P_2$	$\forall X \forall Y, \mathcal{P}_1(X, Y) \Rightarrow \mathcal{P}_2(X, Y)$
c. Typage de domaine d'une propriété	$\exists \mathcal{P}_1:P \sqsubseteq \mathcal{P}_2:C$	$\forall X \forall Y, \mathcal{P}_1:(X, Y) \Rightarrow \mathcal{P}_2:C(X)$
d. Typage de co-domaine d'une propriété	$\exists \mathcal{P}_1:P^- \sqsubseteq \mathcal{P}_2:C$	$\forall X \forall Y, \mathcal{P}_1:(X, Y) \Rightarrow \mathcal{P}_2:C(Y)$

TAB. 2 – Mappings

L'ensemble des connaissances d'un pair, ainsi représentées, est ensuite traduit en clauses propositionnelles. Cet encodage permet de raisonner sur les connaissances du PDMS en réduisant le problème du calcul des réponses à des requêtes conjonctives à un calcul de conséquences et permet d'utiliser SomeWhere implémentant l'algorithme DEcentralized Consequence finding Algorithm (Adjiman *et al.*, 2004).

3 Exemple

Nous décrivons, dans cette section, un exemple de système de gestion de données faisant intervenir deux pairs, \mathcal{P}_1 et \mathcal{P}_2 , pour lesquels les ontologies (cf. TAB. 3) et les mappings (cf. TAB. 4) sont donnés.

\mathcal{P}_1	\mathcal{P}_2
$\forall X, \mathcal{P}_1:Painter(X) \Rightarrow \mathcal{P}_1:Artist(X)$	$\forall X \forall Y, \mathcal{P}_2:Sculpts(X, Y) \Rightarrow \mathcal{P}_2:Sculptor(X)$
$\forall X, \mathcal{P}_1:Painting(X) \Rightarrow \mathcal{P}_1:Artifact(X)$	$\forall X \forall Y, \mathcal{P}_2:Sculpts(X, Y) \Rightarrow \mathcal{P}_2:Sculpture(Y)$
$\forall X \forall Y, \mathcal{P}_1:Paints(X, Y) \Rightarrow \mathcal{P}_1:Creates(X, Y)$	$\forall X \forall Y, \mathcal{P}_2:Refersto(X, Y) \Rightarrow \mathcal{P}_2:Sculptor(X)$
$\forall X \forall Y, \mathcal{P}_1:Paints(X, Y) \Rightarrow \mathcal{P}_1:Painter(X)$	$\forall X, \mathcal{P}_2:Refersto(X, Y) \Rightarrow \mathcal{P}_2:Movement(Y)$
$\forall X \forall Y, \mathcal{P}_1:Paints(X, Y) \Rightarrow \mathcal{P}_1:Painting(Y)$	$\forall X, \mathcal{P}_2:SteelSculptor(X) \Rightarrow \mathcal{P}_2:Sculptor(X)$
$\forall X \forall Y, \mathcal{P}_1:Creates(X, Y) \Rightarrow \mathcal{P}_1:Artist(X)$	$\forall X, \mathcal{P}_2:GlassSculptor(X) \Rightarrow \mathcal{P}_2:Sculptor(X)$
$\forall X \forall Y, \mathcal{P}_1:Creates(X, Y) \Rightarrow \mathcal{P}_1:Artifact(Y)$	$\forall X, \mathcal{P}_2:WoodSculptor(X) \Rightarrow \mathcal{P}_2:Sculptor(X)$
$\forall X \forall Y, \mathcal{P}_1:Belongsto(X, Y) \Rightarrow \mathcal{P}_1:Artifact(X)$	

 TAB. 3 – Ontologies de \mathcal{P}_1 et \mathcal{P}_2

\mathcal{P}_1	\mathcal{P}_2
$\forall X \forall Y, \mathcal{P}_1:Belongsto(X, Y) \Rightarrow \mathcal{P}_2:Movement(Y)$	$\forall X, \mathcal{P}_2:Sculptor(X) \Rightarrow \mathcal{P}_1:Artist(X)$
$\forall X \forall Y, \mathcal{P}_2:Sculpts(X, Y) \Rightarrow \mathcal{P}_1:Creates(X, Y)$	

 TAB. 4 – Mappings de \mathcal{P}_1 et \mathcal{P}_2

Chacun de ces deux pairs peut être sollicité à l'aide de requêtes exprimées avec son propre vocabulaire. Ainsi, si un utilisateur s'adresse au pair \mathcal{P}_1 pour connaître l'ensemble des artefacts connus du système, la requête suivante sera posée : $Q_1(X) \equiv \mathcal{P}_1:Artifact(X)$. Le calcul des réponses aux requêtes se fait en deux temps. La requête Q_1 est d'abord réécrite en un ensemble de requêtes subsumant Q_1 . Une réécriture possible de $Q_1(X) \equiv \mathcal{P}_1:Artifact(X)$ est, par exemple, $\mathcal{P}_4:Painting(X)$, ce qui signifie qu'un moyen d'obtenir des instances de $Artifact(X)$ consiste à aller rechercher les instances de $Painting(X)$ au sein du

même pair. Une autre réécriture est $\exists Y, \mathcal{P}_2:Belongsto(X, Y)$, ce qui signifie qu'un autre moyen d'obtenir des instances de $Artifact(X)$ consiste à retenir les valeurs de X des instances de $Belongsto(X, Y)$ dans \mathcal{P}_1 également.

Ces réécritures sont ensuite évaluées afin d'obtenir les données associées. Sur cet exemple, les réécritures obtenues sont :

- $R_1(X) \equiv \mathcal{P}_1:Artifact(X)$
- $R_2(X) \equiv \mathcal{P}_1:Painting(X)$
- $R_3(X) \equiv \exists Y, \mathcal{P}_1:Belongsto(X, Y)$
- $R_4(X) \equiv \exists Y, \mathcal{P}_1:Paints(Y, X)$
- $R_5(X) \equiv \exists Y, \mathcal{P}_1:Creates(Y, X)$
- $R_6(X) \equiv \exists Y, \mathcal{P}_2:Sculpts(Y, X)$
- $R_7(X) \equiv \mathcal{P}_2:Sculpture(X)$

La réécriture R_2 est obtenue par \mathcal{P}_1 en utilisant l'inclusion de classes, tandis que R_4 et R_5 sont obtenues en utilisant le typage de co-domaine des propriétés $\mathcal{P}_1:Paints$ et $\mathcal{P}_1:Creates$. Les mappings $\forall X \forall Y, \mathcal{P}_1:Belongsto(X, Y) \Rightarrow \mathcal{P}_2:Movement(Y)$ et $\forall X \forall Y, \mathcal{P}_2:Sculpts(X, Y) \Rightarrow \mathcal{P}_1:Creates(X, Y)$ permettent d'obtenir les réécritures R_3 et R_6 . La requête $Q_1(X)$ est ensuite propagée au pair \mathcal{P}_2 du fait de l'existence de mappings avec des relations de \mathcal{P}_2 . Ce dernier transmettra à \mathcal{P}_1 les réécritures qu'il obtient, i.e. R_7 . La réponse à la requête est obtenue après évaluation de chacune des requêtes réécrites.

4 Spécificités de la découverte de correspondances dans le cadre de PDMS

4.1 Nombre d'éléments à mettre en correspondance indéterminé

Dans une architecture centralisée, la découverte de correspondances entre deux ontologies revient à faire le produit cartésien des différents éléments de ces ontologies suivant un ou plusieurs calculs de distances, puis à analyser les mesures de similarité obtenues. Dans le cadre des PDMS, le nombre de paires, a priori important et inconnu, rend impossible ce type de calcul. Il est nécessaire de définir les techniques de filtrage sélectionnant les concepts pour lesquels il serait intéressant d'établir des mappings.

4.2 Importance de la localisation des connaissances

Un pair n'a accès qu'à une partie des connaissances disponibles sur le PDMS. Il ne connaît que les éléments qu'il a définis. Dans un cadre totalement distribué, comme c'est le cas dans SomeRDFS, la gestion du réseau est totalement décentralisée, aucun pair n'a une vision globale des connaissances du PDMS. Cette vision partielle des connaissances influe sur les mappings qui peuvent être découverts automatiquement et conditionne les mécanismes à mettre en œuvre pour les découvrir.

4.3 Exploitation du raisonnement

Le raisonnement effectué dans un PDMS est une source de richesse pour découvrir des mappings entre ontologies. L'analyse des réponses obtenues à une requête posée à un pair peut, par exemple, permettre de s'apercevoir de l'absence de participation d'autres pairs et donc de connaître les concepts pour lesquels il serait intéressant de rechercher des mappings si on souhaite des réponses plus larges.

5 Exploitation des réponses aux requêtes

Les réponses aux requêtes sont exploitables de deux façons différentes. D'une part, elles permettent de découvrir des raccourcis de mappings, d'autre part, elles permettent d'identifier les éléments qu'il serait intéressant de mettre en correspondance.

5.1 Découverte de raccourcis de mappings

Les raccourcis de mapping renforcent le réseau en créant des liens directs entre des relations de deux pairs différents là où jusqu'alors n'existaient, dans le PDMS, que des liens indirects. Ainsi dans FIG. 1, la spécialisation de *Artist* par *Pianist* ($\mathcal{P}_1:Pianist(X) \Rightarrow \mathcal{P}_2:Artist(X)$) est un raccourci de mapping déductible de $\mathcal{P}_1:Pianist(X) \Rightarrow \mathcal{P}_3:Musician(X)$ et de $\mathcal{P}_3:Musician(X) \Rightarrow \mathcal{P}_2:Artist(X)$.

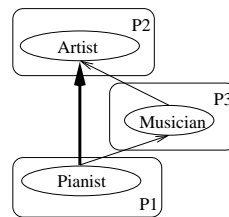


FIG. 1 – Exemple de raccourci de mapping

L'objectif n'est pas de rajouter ce type de mapping systématiquement mais de façon sélective. En effet ces mappings peuvent être intéressants à représenter car, bien qu'ils ne permettent pas d'aboutir à des réponses plus riches, ils peuvent être très utiles en cas de disparition de pairs du PDMS. Dans FIG. 1, les trois concepts appartiennent chacun à un pair différent. Si \mathcal{P}_3 se déconnecte, en l'absence de raccourci du mapping ($\mathcal{P}_1:Pianist(X) \Rightarrow \mathcal{P}_2:Artist(X)$), \mathcal{P}_1 perdra l'accès aux données de \mathcal{P}_2 et de \mathcal{P}_3 .

La détermination des raccourcis de mappings peut être automatique. Nous proposons, pour cela, d'exploiter les résultats des requêtes posées dans SomeRDFS et d'appliquer ensuite des techniques de sélection des raccourcis pertinents à représenter.

5.1.1 L'exploitation des résultats des requêtes posées dans SomeRDFS

L'analyse des interactions entre les utilisateurs et les pairs peut permettre de comprendre ce que recherchent précisément ces utilisateurs. Dans l'environnement SomeRDFS, l'obtention de données recherchées s'effectue en deux étapes : (1) le calcul des réécritures maximales de chaque atome de la requête posée, (2) l'évaluation des réécritures. Le calcul des réécritures maximales de chaque atome de la requête fournit un ensemble de conjonctions de relations (classes ou propriétés). Les réécritures peuvent être locales. Leurs relations appartiennent toutes au vocabulaire du pair interrogé. Les réécritures peuvent être distantes. Leurs relations appartiennent toutes à un même pair mais il est différent du pair interrogé. Enfin, les réécritures peuvent être obtenues par intégration. Leurs relations appartiennent à des pairs différents. La phase d'évaluation consiste à fournir les données instanciant les requêtes constituant les réécritures. Ces deux étapes peuvent être, ou non, dissociées. L'approche que nous proposons pour découvrir les raccourcis de mappings pertinents à représenter est basée sur la dissociation de ces deux étapes.

La dissociation de ces deux étapes est intéressante lorsqu'un utilisateur ne trouve pas, au sein du vocabulaire du pair auquel il s'adresse, la classe lui permettant de définir précisément les données qu'il recherche, et qu'il est dans l'obligation d'indiquer une autre classe. Cette autre classe peut être plus spécifique. Dans ce cas, toutes les réponses attendues ne seront pas obtenues. Elle peut être plus générale. Dans cet autre cas, elle permettra d'obtenir toutes les réponses attendues par l'utilisateur mais contiendra, en revanche, également des réponses inutiles. Ainsi, par exemple, si l'utilisateur s'adresse au pair \mathcal{P}_1 dans l'espoir d'obtenir les données de la classe *SteelSculptor*, il peut, en l'absence de la classe *SteelSculptor* dans \mathcal{P}_1 , s'intéresser aux données d'une classe plus générale en posant la requête $Q_2(X) \equiv \mathcal{P}_1:Artist(X)$. Il obtiendra, parmi les réécritures, $\mathcal{P}_2:SteelSculptor(X)$ indiquant que les données instanciant *SteelSculptor(X)* sont des artistes mais également $\mathcal{P}_2:WoodSculptor(X)$ ou $\mathcal{P}_2:GlassSculptor(X)$ dont il n'est pas utile de rechercher les données, compte tenu du besoin précis de l'utilisateur. Le fait de dissocier le calcul des réécritures de leur évaluation est, dans ce cas, intéressant. Il permet à l'utilisateur de sélectionner les réécritures pour lesquelles il demande l'évaluation.

5.1.2 Technique des sélections des raccourcis pertinents à représenter

La trace des interactions entre les utilisateurs et les pairs peut être analysée pour trouver les raccourcis pertinents à représenter. Nous proposons d'ajouter des raccourcis de mappings lorsqu'il s'agit de liens directs avec des classes intéressantes pour les utilisateurs. Ces classes doivent appartenir au vocabulaire de pairs distants. Elles sont plus spécifiques que celles du pair étudié. Elles n'apparaissent pas, pour l'instant, dans ses mappings, mais sont apparues dans des réécritures et les utilisateurs ont, à plusieurs reprises, demandé leur évaluation.

Lorsqu'il s'agit d'une seule classe nommée, il s'agira d'ajouter un mapping

établissant un lien de spécialisation entre cette classe et la classe (plus générale) qui a fait l'objet de la requête initiale. Ce mapping est ajouté au pair interrogé. Dans l'exemple précédent, il s'agira d'ajouter $\mathcal{P}_2:SteelSculptor(X) \Rightarrow \mathcal{P}_1:Artist(X)$ dans \mathcal{P}_1 .

Lorsqu'il s'agit de l'union de plusieurs classes nommées, plusieurs mappings construits de la même façon que précédemment seront ajoutés. Ainsi si l'utilisateur recherche des données du type *MineralSculptor* en s'adressant à \mathcal{P}_1 , cette classe n'appartenant pas au vocabulaire du pair interrogé, la requête portera sur $\mathcal{P}_1:Artist(X)$ considérée comme plus générale. Si l'utilisateur demande, après avoir eu connaissance de toutes les réécritures, l'évaluation de $\mathcal{P}_1:SteelSculptor(X)$ et de $\mathcal{P}_2:GlassSculptor(X)$, les deux mappings suivants seront ajoutés : $\mathcal{P}_2:SteelSculptor(X) \Rightarrow \mathcal{P}_1:Artist(X)$ et $\mathcal{P}_2:GlassSculptor(X) \Rightarrow \mathcal{P}_1:Artist(X)$.

L'ajout de ces mappings dépend du nombre de fois que, suite à une requête posée, l'évaluation de classes distantes est demandée. Ce nombre n'a pas, pour l'instant, été fixé. Il sera précisé, lorsque des expérimentations auront été réalisées.

5.2 Identification de classes cibles

Le mécanisme de réécriture mis en œuvre dans SomeRDFS renvoie un ensemble de conjonctions de relations (classes, propriétés). L'intérêt principal des mappings est de permettre la propagation de ce mécanisme à des pairs distants. Ceci leur permet de contribuer aux réponses fournies à des requêtes posées à d'autres pairs.

Lorsqu'un utilisateur s'adresse à un pair, il arrive toutefois que les réponses proviennent toutes de ce pair. Cette situation révèle un manque de mappings de spécialisation entre les relations de ce pair et celles de pairs distants. L'identification de ce phénomène est possible par analyse des réponses obtenues aux requêtes, ou, plus précisément, par observation de la localisation des éléments qui composent les réponses. Supposons, par exemple, que la requête $Q_3(X) \equiv \mathcal{P}_1:Painter(X)$ soit posée à \mathcal{P}_1 et que la réécriture obtenue soit la suivante : $\exists Y, P_1:Paints(X, Y)$. Dans cet exemple, nous constatons que seul \mathcal{P}_1 participe à la réponse à Q_3 . Ceci signifie qu'il n'existe aucun mapping de spécialisation ou de typage concernant $\mathcal{P}_1:Painter(X)$.

Une étude nous a permis de définir deux cas d'utilisation pour lesquelles le calcul de réécritures est un indice pour trouver des classes appartenant à des pairs distants qui peuvent être rapprochées de classes du pair interrogé. Nous présentons successivement ces deux cas d'utilisation.

Cas d'utilisation 1 :

Soient les pairs \mathcal{P}_1 , \mathcal{P}_2 et \mathcal{P}_3 contenant respectivement les classes C_1 , C_2 et C_3 et les mappings suivants : $\mathcal{P}_1:C_1(X) \Rightarrow \mathcal{P}_2:C_2(X)$ et $\mathcal{P}_3:C_3(X) \Rightarrow \mathcal{P}_2:C_2(X)$ représentés dans les deux pairs concernés pour chacun.

1. L'utilisateur s'adresse à \mathcal{P}_3 et pose la requête $Q_4(X) \equiv \mathcal{P}_3:C_3(X)$
2. SomeRDFS ne fournit aucune réécriture.

Cas d'utilisation 2 :

Soient le pair \mathcal{P}_1 contenant la classe C_1 et le pair \mathcal{P}_2 contenant les classes C_2 et C_3 telles que $\mathcal{P}_2:C_2(X) \Rightarrow \mathcal{P}_2:C_3(X)$. Nous supposons que le mapping $\mathcal{P}_2:C_2(X) \Rightarrow \mathcal{P}_1:C_1(X)$ est représenté à la fois dans \mathcal{P}_1 et dans \mathcal{P}_2 .

1. L'utilisateur s'adresse à \mathcal{P}_2 et pose la requête $Q_5(X) \equiv \mathcal{P}_2:C_3(X)$
2. SomeRDFS fournit la réécriture suivante : $\mathcal{P}_2:C_2(X)$. Aucune réécriture composée de relations de pairs distants n'est donnée.

C_2 et C_3 sont respectivement appelées classes cibles car c'est à partir d'elles que des mises en correspondance intéressantes peuvent être trouvées. Nous décrivons en section 6 le mode de découverte de mappings qui s'appuie sur l'identification de ces classes cibles.

6 Découverte de mappings nouveaux

Nous proposons une procédure de découverte de mappings nouveaux qui s'appuie sur les raisonnements effectués au sein de SomeRDFS en identifiant des classes à partir desquelles des mises en correspondance intéressantes peuvent être trouvées. Ces classes, appelées classes cibles sont notées C_{CIBLE} . Sans liens avec d'autres pairs via des mappings, ces classes ne permettent pas de propager le raisonnement vers un autre pair. Pour chaque classe cible, nous nous proposons alors de déterminer, dans un second temps, un ensemble des classes entre lesquelles il serait pertinent de rechercher des mises en correspondance. Cet ensemble de classes sera appelé contexte de mappings et noté CM . Le processus de découverte de mappings nouveaux est donc un processus en deux étapes : (1) recherche des classes cibles, (2) recherche des contextes de mappings associés.

Notre approche est basée sur l'idée selon laquelle il est pertinent de rechercher des mises en correspondance entre des classes ayant des points communs. Dans le cadre de SomeRDFS, les points communs considérés seront (1) l'existence d'une classe commune plus générale, ou bien (2) l'existence d'une classe commune plus spécifique. Nous proposons un processus de recherche de contextes de mappings pour chacune de ces situations.

Nous nous basons sur l'existence d'une classe commune plus générale lorsque C_{CIBLE} n'a qu'une classe C_g (au sein du même pair ou dans la définition d'un mapping) qui la généralise. Dans ce cas, nous proposons de poser une requête sur cette classe ($Q(X) \equiv C_g(X)$) et d'en calculer les réécritures. Ces dernières fournissent un ensemble de classes plus spécifiques que C_g entre lesquelles il est pertinent de rechercher l'existence de correspondances. Cet ensemble constitue CM . Si $\exists C_g/C_{CIBLE} \Rightarrow C_g$ et $Q(X) \equiv C_g(X)$ alors $CM =$ l'ensemble des classes correspondant à des réécritures de $Q(X)$.

Nous nous basons sur l'existence d'une classe commune plus spécifique lorsque C_{CIBLE} a plusieurs classes C_g (au sein du même pair ou dans la définition

d'un mapping) qui la généralisent. Dans ce cas, l'ensemble de ces généralisants constitue un contexte de mappings CM . Si $G = \{C_g/C_{CIBLE} \Rightarrow C_g\}$ avec $G = G_1 \cup G_2/G_1 = \{C_g \in P_{CIBLE}\}$ et $G_2 = \{C_g \notin P_{CIBLE}\}$ alors $CM = G$.

Ces deux situations correspondent aux cas 1 et 2 décrits dans la section précédente. Nous reprenons ces deux cas en les complétant par les étapes menant à la découverte de mises en correspondance.

Cas d'utilisation 1 :

1. L'utilisateur s'adresse à \mathcal{P}_3 et pose la requête $Q_4(X) \equiv \mathcal{P}_3:C_3(X)$
2. SomeRDFS ne fournit aucune réécriture
3. L'utilisateur pose la requête $Q_2(X) \equiv \mathcal{P}_2:C_2(X)$ suite à l'inexistence de réécriture de Q_4 et après avoir pris connaissance du mapping $\mathcal{P}_3:C_3(X) \Rightarrow \mathcal{P}_2:C_2(X)$ dans \mathcal{P}_3
4. SomeRDFS fournit les réécritures suivantes : $\mathcal{P}_1:C_1(X), \mathcal{P}_3:C_3(X)$
5. L'utilisateur débute une procédure de recherche de mappings $CM = \{\mathcal{P}_1:C_1(X) \text{ et } \mathcal{P}_3:C_3(X)\}$

Cas d'utilisation 2 :

1. L'utilisateur s'adresse à \mathcal{P}_2 et pose la requête $Q_5(X) \equiv \mathcal{P}_2:C_3(X)$
2. SomeRDFS fournit la réécriture suivante : $\mathcal{P}_2:C_2(X)$. Aucune réécriture composée de relations de paires distants n'est donnée.
3. L'utilisateur débute une procédure de recherche de mappings du fait qu'il connaît l'existence de $\mathcal{P}_2:C_2(X) \Rightarrow \mathcal{P}_1:C_1(X)$. $CM = \{\mathcal{P}_1:C_1(X) \text{ et } \mathcal{P}_2:C_3(X)\}$

La recherche de mappings s'effectue entre les éléments de l'ensemble CM . Cet ensemble comprend des relations de différents paires, celui à qui la requête a été posée pour obtenir cet ensemble, et les paires distants ayant contribué à la réponse à la requête via des réécritures. Le pair interrogé a une certaine compréhension des classes de CM faisant partie de son vocabulaire puisque celles-ci font partie de son ontologie. En revanche, les classes de CM appartenant au vocabulaire de paires distants correspondent à des noms de classes isolées. Le problème d'alignement qui se pose consiste à mettre en correspondance des classes prises isolément avec une autre classe appartenant à une taxonomie. Des techniques de type terminologiques ou basées sur l'utilisation de ressources terminologiques complémentaires sont utilisables. En revanche, les techniques structurelles communément utilisées, consistant à comparer la structure de deux ontologies, sont inadaptées.

Les techniques terminologiques ont souvent recours au calcul de distances. Ce calcul peut permettre d'appréhender un grand nombre de cas mais il est inopérant lorsque les termes à rapprocher sont des synonymes. Le traitement des synonymes peut être fait en utilisant un thesaurus, tel Wordnet, à condition d'être en mesure de définir le contexte des différents rapprochements effectués.

Beaucoup de travaux ont été réalisés (Kalfoglou & Schorlemmer, 2003; Shvaiko & Euzenat, 2005) en alignement d'ontologies ces dernières années mais les problèmes que nous soulevons sont particuliers et n'ont pas, à notre connaissance, été résolus.

7 Conclusion

Au travers de ce travail préliminaire, qui s'inscrit dans le cadre de systèmes P2P, nous avons étudié comment tirer parti du processus de raisonnement logique mis en oeuvre dans SomeRDFS, dans le but d'aider à la découverte de correspondances entre ontologies. Notre étude a montré qu'il s'agissait d'un outil utile pour orienter la découverte de nouveaux mappings. Elle nous a permis de proposer une méthode de découverte de mappings combinant mise en oeuvre de mécanismes logiques et application de techniques d'alignement. Les techniques d'alignement usuelles ne sont pas appropriées. Les perspectives à ce travail consistent à approfondir notre étude puis à concevoir et implémenter des techniques particulièrement adaptées à ce contexte d'alignement en prenant appui sur des travaux réalisés dans l'équipe en alignement de taxonomies (Reynaud & Safar, 2007; Kefi *et al.*, 2006).

Références

- ADJIMAN P., CHATALIC P., GOASDOUÉ F., ROUSSET M.-C. & SIMON L. (2004). Distributed reasoning in a peer-to-peer setting. In *European Conference on Artificial Intelligence*, p. 945–946.
- ADJIMAN P., GOASDOUÉ F. & ROUSSET M.-C. (2006). Somerdfs in the semantic web. *Journal on Data Semantics*, **8**, p. 158–181.
- KALFOGLOU Y. & SCHORLEMMER M. (2003). Ontology mapping: the state of the art. *The Knowledge Engineering Review*, **18**, p. 1–31.
- KEFI H., SAFAR B. & REYNAUD C. (2006). Alignement de taxonomies pour l'intégration de sources d'information heterogenes. In *Reconnaissances des Formes et Intelligence Artificielle*.
- REYNAUD C. & SAFAR B. (2007). Techniques structurelles d'alignement pour portail web. In *Numéro special fouille du Web, Revue RNTI à paraître*.
- SHVAIKO P. & EUZENAT J. (2005). A survey of schema-based matching approaches. *Journal of Data Semantics IV*, p. 146–171.