Comment fusionner des ontologies avec la réécriture de graphes ?

Mariem Mahfoudh* — **Germain Forestier*** — **Laurent Thiry*** — **Michel Hassenforder***

* *Université de Haute-Alsace, MIPS EA 2332*
12 rue des Frères Lumière, 68093 Mulhouse, France
{mariem.mahfoudh, germain.forestier, laurent.thiry, michel.hassenforder}@uha.fr

RÉSUMÉ. *Au cours de ces dernières années, les ontologies se sont imposées comme un outil incontournable de représentation des connaissances. Cette popularité a conduit au développement d'ontologies similaires ou partiellement redondantes. Cette multiplication de ressources disponibles a mené aux études traitant de leur réutilisation et/ou leur fusion. Dans ce contexte, nous proposons une nouvelle approche de fusion d'ontologies basée sur les règles de réécriture de graphes. A l'aide du domaine des grammaires de graphes typés, nous avons défini un nouveau formalisme capable de représenter des ontologies et de produire le résultat de leur fusion. Pour valider notre proposition, l'approche a été implémentée et testée sur une dizaine d'ontologies dans différents domaines d'application.*

ABSTRACT. *In the recent years, ontologies have emerged as an essential formalism for knowledge representation. This popularity has led to the development of similar or partially redundant ontologies. Thus, this multiplication of resources has led to study their reuse and/or their merge. In this context, we propose a new approach for merging ontologies based on graph rewriting rules. Using the domain of typed graph grammars, we defined a new formalism able to represent ontologies and produce the result of their merge. To validate our proposal, the approach has been implemented and tested on many ontologies in different application domains.*

MOTS-CLÉS : *Fusion d'ontologies, grammaires de graphes typés, règles de réécriture de graphes, GROM.*

KEYWORDS: *Merging ontologies, typed graphs grammars, rewriting rules, GROM.*

1. Introduction

Appréciées pour leur fonction de représentation explicite, formelle et sémantique des connaissances, les ontologies sont largement utilisées dans de nombreux domaines : web sémantique (Mahfoudh *et al.*, 2011), médical (Forestier *et al.*, 2011), entreprises créatives (Thiry *et al.*, 2014), routier (Châabane *et al.*, 2008), etc. Cette popularité a conduit au développement de plusieurs ontologies similaires ou partiellement redondantes modélisant des domaines identiques ou connexes. Ainsi, cette multiplication de ressources a mené à l'étude de leur fusion pour une meilleure réutilisation.

Selon (Klein, 2001), la fusion d'ontologies est *"la création d'une nouvelle ontologie à partir de deux ou plusieurs ontologies existantes avec des parties qui se chevauchent"*. La création de la *nouvelle ontologie* (aussi appelée l'ontologie globale) se passe essentiellement par deux grandes phases : 1) la recherche de correspondances entre les concepts des ontologies afin de les aligner et 2) la fusion d'ontologies se basant sur l'alignement défini (Raunich *et al.*, 2014). Pour accomplir la première phase, plusieurs techniques de similarité ont été proposées dans la littérature. Elles sont généralement classées en quatre catégories : 1) les techniques lexicales, 2) les techniques structurelles, 3) les techniques sémantiques et 4) la combinaison des différentes techniques précédentes. Ainsi, les techniques lexicales considèrent les noms des entités et les comparent comme des chaînes de caractères, ex. la distance de Levenshtein (Levenshtein, 1966). Les techniques structurelles considèrent la structure des ontologies pour détecter les relations de subsomption, ex. Children et Leaves (Do *et al.*, 2002). Quant aux techniques sémantiques, elles cherchent à identifier les relations sémantiques entre les concepts en utilisent des sources externes, ex. les ontologies linguistiques et les dictionnaires. En effet, la recherche de correspondances entre ontologies est un axe de recherche largement étudié qui présente des résultats de plus en plus importants (Pavel *et al.*, 2013). Quant à la phase de fusion d'ontologies, c'est là où peu de travaux existent car celle ci présente un verrou scientifique important. Plusieurs questions se sont toujours posées. Comment fusionner des ontologies ? Quel cadre formel peut-on utiliser pour suivre ce processus ? Quelle est la "meilleure" stratégie de fusion "symétrique" ou bien "asymétrique" ? etc. Ainsi, pour répondre à ces questions, nous proposons dans cet article l'utilisation de réécriture de graphes. L'idée principale de la réécriture de graphes, également appelée grammaires de graphes, est la modification de graphes via des règles de réécriture. Elle permet de décrire la manière de transformer un graphe en un autre tout en précisant quand et comment faire les changements. Ainsi, notre objectif est de coupler le domaine des grammaires de graphes et les ontologies afin de définir une approche formelle de fusion d'ontologies aidant à remédier aux problèmes de leur intégration.

Le reste de l'article est organisé comme suit : la section 2 rappelle les concepts fondamentaux pour comprendre ce que sont les réécritures de graphes. La section 3 présente un modèle de représentation des ontologies suivant ce formalisme. La section 4 décrit notre approche de fusion d'ontologies. La section 5 donne une implémentation des éléments proposés. La section 6 présente des travaux connexes. Enfin, une conclusion synthétise le travail présenté et donne les perspectives envisagées.

2. Réécriture de graphes

Cette section présente certaines définitions de base concernant les fondements théoriques de la réécriture de graphes.

Graphe. Un graphe $G(N, E)$ est une structure composée par un ensemble de nœuds (N), d'arêtes (E) et d'une application $s : E \rightarrow N \times N$ qui attache les nœuds source/destination à chaque arête.

Graphe attribué. Un graphe attribué est un graphe étendu par un ensemble d'attributs A, une fonction d'attribution $att : N \cup E \rightarrow \mathcal{P}(A)$, $\mathcal{P}$ représentant l'ensemble des parties, et une fonction d'évaluation $val : A \rightarrow V$. Ainsi, chaque nœud ou arrête peut avoir un ensemble d'attributs ($\mathcal{P}(A)$) dont les valeurs seront données par val.

Morphisme de graphes. Soient deux graphes $G(N, E)$ et $G'(N', E')$, un morphisme de graphes $m(f, g)$ est une application de G à G' définie par deux applications $f : N \rightarrow N'$ et $g : E \rightarrow E'$. Un morphisme doit préserver la structure, c.à.d si $e = (s, t)$ et $g(e) = e' = (s', t')$ alors $s' = f(s)$ et $t' = f(t)$.

Typage. Le typage est un morphisme d'un graphe $G(N, E)$ à un graphe type $TG(N_T, E_T)$ avec N_T correspond aux types des nœuds et E_T aux types des arêtes.

Grammaires de graphes typés. Une grammaire de graphe typé est une structure mathématique définie par $TGG = (G, TG, P)$ avec :

– G est un graphe initial, appelé aussi graphe hôte ;

– TG est un graphe type précisant le type de l'information représentée dans le graphe hôte (type des nœuds et des arêtes) ;

– P un ensemble de règles de réécriture, appelées aussi règles de production ou de transformations de graphes. Une règle de réécriture r est une paire de graphes pattern (LHS, RHS) avec : 1) LHS (Left Hand Side) représente la pré-condition de la règle de réécriture et décrit la structure qu'il faut trouver dans un graphe G pour pouvoir appliquer la règle ; 2) RHS (Right Hand Side) représente la post-condition de la règle de réécriture et doit remplacer LHS dans G. À noter que les règles peuvent également avoir des conditions supplémentaires appelées NAC (Negative Application Conditions). Ce sont des graphes pattern définissant des conditions qui ne doivent pas être vérifiées pour que la règle de réécriture puisse être appliquée.

La transformation de graphe consiste ainsi à définir comment un graphe G peut être transformé en un nouveau graphe G'. Cette transformation peut être réalisée selon deux types d'approches (Rozenberg, 1999) : les approches ensemblistes (Node replacement, Edge replacement, etc.) et les approches algébriques. Dans ce travail, nous utilisons les approches algébriques basées sur le concept de *pushout* de la théorie des catégories (Barr *et al.*, 1990).

Théorie des catégories. Une catégorie est une structure composée de : 1) une collection d'objets O ; 2) un ensemble de morphismes M et une fonction $s : M \rightarrow O \times O$, $s(f) = (A, B)$ est notée alors $f : A \rightarrow B$; 3) une loi de composition $(\circ) : M \times M \rightarrow M$; 4) un morphisme identité pour chaque objet $id : O \rightarrow O$. La loi de composition doit être associative et avoir l'identité comme élément neutre. Comme exemple, dans la catégorie *"Graph"*, O représente l'ensemble des graphes et M les morphismes de

graphes. Les éléments présentés dans la suite s'appuient sur cette catégorie.
Pushout. Soient trois objets de la catégorie des graphes : G_1, G_2 et G_3 et deux morphismes $f : G_1 \rightarrow G_2$ et $g : G_1 \rightarrow G_3$. Le pushout de G_2 et G_3 consiste à : 1) un objet G_4 et deux morphismes $f' : G_2 \rightarrow G_4$ et $g' : G_3 \rightarrow G_4$ avec $f' \circ f = g' \circ g$; 2) pour tout morphisme $f'' : G_2 \rightarrow X$ et $g'' : G_3 \rightarrow X$ tel que $f \circ f'' = g \circ g''$, il y a un seul morphisme $k : G_4 \rightarrow X$ tel que $f' \circ k = f''$ and $g' \circ k = g''$.
A partir de là, deux variantes sont proposées pour la réécriture des graphes : le *Simple pushout SPO* (Löwe, 1993) et le *Double poushout DPO* (Ehrig, 1979). Dans ce travail, seule l'approche SPO a été considérée car elle se voit plus générale. Une comparaison détaillée des deux approches est proposée par (Ehrig *et al.*, 1997). Ainsi, appliquer une règle de réécriture à un graphe initial G, selon la méthode SPO, revient à : 1) trouver un morphisme (m) permettant d'identifier un sous graphe de graphe G qui correspond (match) avec la partie LHS (m : $LHS \rightarrow G$) ; 2) appliquer la règle de réécriture sur le sous graphe en le remplaçant par $m(RHS)$ et supprimant les arêtes suspendues, i.e. les arêtes qui ont une extrémité non liée à un nœud. Ainsi, d'une manière générale, nous avons $SPO(G, LHS, RHS) = G'$ (voir Figure 1).

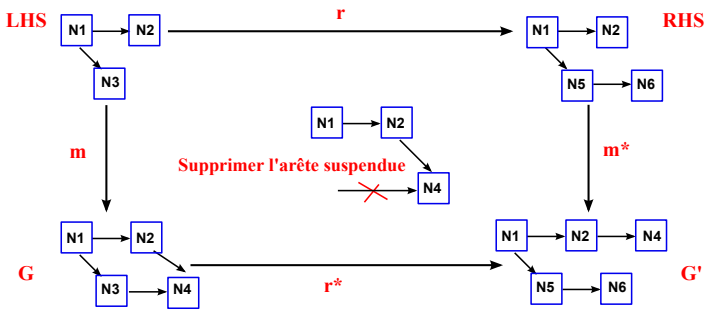

Figure 1. *Exemple d'application d'une règle de réécriture avec l'approche SPO.*

3. Modèle de transformation de graphes pour la réutilisation d'ontologies

Les langages de représentation des ontologies sont principalement basés sur le modèle RDF[1] (Resource Description Framework) qui est également basé sur des graphes. Par conséquent, représenter des ontologies sous forme de graphes attribués est tout à fait cohérent et approprié.

Proposition 1. Nous considérons une ontologie comme un graphe hôte G possédant une relation de typage avec le graphe type (TG), où TG représente le méta-modèle de l'ontologie. Pour être conforme aux standards, c'est OWL[2] qui a été retenu comme méta-modèle d'ontologies. Ainsi, les types des nœuds considérés sont :

1. w3.org/RDF
2. w3.org/TR/owl-ref

$$N_T = \{Class(C), Property(P), ObjectProperty(OP), DataProperty(DP),\text{-}\\ Individual(I), DataType(D), Restriction(R)\}.$$

Les types des arêtes correspondent aux axiomes utilisés pour relier les différentes entités :

$$E_T = \{subClassOf, equivalentTo, range, domain, ...\}.$$

La Figure 2 présente deux ontologies sous la forme de leur graphes hôtes, représentées selon le formalisme de grammaire de graphes typés. Par exemple, la première ontologie O_1 est définie par :

– $N = \{C, I, OP, R\}$ avec : 1) $C = \{"Person", "European\_Car", "BMW", "Fiat", ...\}$; 2) $I = \{"John", "FiatPunto"\}$; 3) $OP = \{"has\_Owner"\}$; 4) $R = \{"CardinalityRestriction"\}$, la restriction est du type max et définie sur la propriété "has_Owner" ce qui signifie que chaque "Automobile" doit avoir au plus un seul propriétaire "Person".

– $E = \{subClassOf, memberOf, range, domain, objectPropertyAssertion, hasRestriction, onProperty\}$.

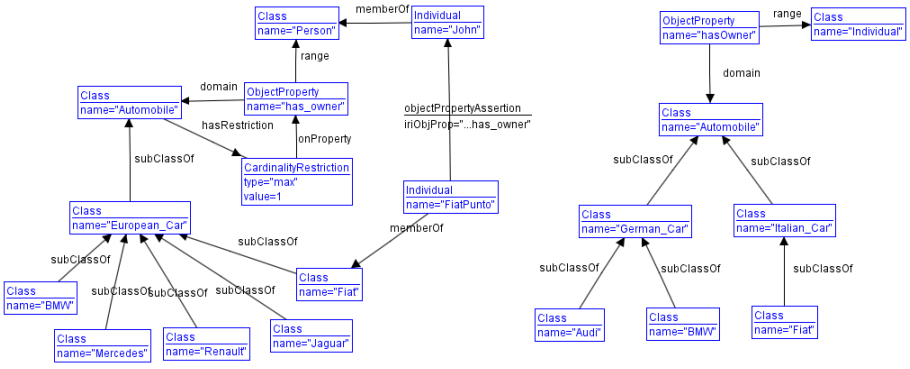

Figure 2. *Exemple des graphes hôtes : (gauche) ontologie O_1, (droite) ontologie O_2.*

Proposition 2. Avec la proposition 1, les changements ontologiques peuvent être formalisés par un ensemble de règles de réécriture :

$$r_i = (NAC_i, LHS_i, RHS_i, CHD_i) \text{ avec}\\ i \in \{AddClass, RemoveDataProperty, RenameIndividual, ...\}.$$

Dans cette définition étendue, CHD correspond à l'ensemble des changements dérivés ajoutés à un changement ontologique pour corriger ses éventuelles inconsistances.

La Figure 3 présente la règle de réécriture du changement ontologique $AddIndividual$. Elle permet d'ajouter un individu "Pascal" tout en spécifiant son

type, la classe "Person". La règle assure, grâce au NAC, la non redondance de données, i.e. elle empêche l'application du changement dans le cas où l'individu existe déjà dans l'ontologie. Plus de détails et des exemples de changements ontologiques sont présentés dans notre travail (Mahfoudh *et al.*, 2013).

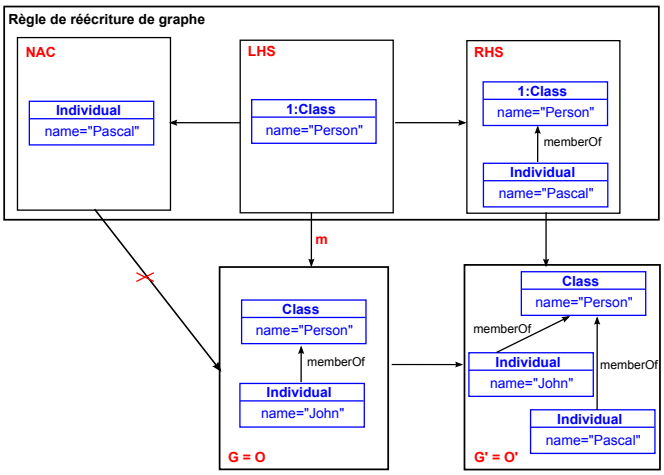

Figure 3. *Règle de réécriture du changement $AddIndividual$.*

4. Fusion d'ontologies avec la réécriture de graphes

Après avoir présenté le modèle de représentation des ontologies à l'aide des grammaires de graphes, nous détaillons dans cette section comment fusionner les ontologies via ce formalisme. Ainsi, avant de commencer le processus de fusion, il est indispensable d'identifier au préalable l'ensemble des correspondances entre les concepts d'ontologies. Dans notre travail, nous nous intéressons à quatre types de correspondances :

1) Nœuds communs entre les deux ontologies possédant des noms identiques, $NC = \{N_i | (N_i \in N(O_1)) \wedge (\exists N_j \in N(O_2) \cdot (N_{iT} = N_{jT}) \wedge (distanceSyntaxique(N_i.name, N_j.name) = 0)\}$. Considérons l'exemple d'ontologies présentées par la Figure 2, $NC = \{"Automobile"; "Fiat; "BMW"\}$.

2) Nœuds équivalents dont les chaines de caractères de leurs noms sont syntaxiquement proche $NE = \{(N_i, N_j) | (N_i \in N(O_1)) \wedge (N_j \in N(O_2)) \wedge (N_{iT} = N_{jT}) \wedge (distanceSyntaxique(N_i.name, N_j.name) < seuil)\}$, ex. $NE = \{("has\_owner", "hasOwner")\}$;

3) Nœuds partageant une relation sémantique de synonymie, $NS = \{(N_i, N_j) | (N_i \in N(O_1)) \wedge (N_j \in N(O_2)) \wedge (N_{iT} = N_{jT}) \wedge$ *(distanceSémantique*$(N_i.name, N_j.name) = 0)\}$, ex. $NS = \{("Person", "Individual")\}$.

4) Nœuds partageant une relation de subsomption, $NIsa = \{(N_i, N_j)|(N_i \in N(O_1)) \wedge (N_j \in N(O_2)) \wedge (N_{iT} = N_{jT}) \wedge (Isa(N_i) = N_j)\}$, ex. $NIsa = \{(\text{"}German\_Car\text{"}, \text{"}European\_Car\text{"}), (\text{"}Italian\_Car\text{"}, \text{"}European\_Car\text{"}), (\text{"}Mercedes\text{"}, \text{"}German\_Car\text{"})\}$.

Le processus de fusion d'ontologies repose sur l'approche SPO et décrit par l'algorithme 1.

Entrées: deux ontologies O_1, O_2
un ensemble de correspondances : NC, NE, NS, $NIsa$
Sorties: une ontologie globale OG

pour *N in* $\{NE\}$ **faire**
  $O'_1 \leftarrow SPO\_RenameEntity$ (O_1, $NE\{O_1\}$, $NE\{O_2\}$);
fin

$\{NC\} \leftarrow \{NC\} \cup NE\{O_1\}$;

$OC \leftarrow$ Créer le graphe de l'ontologie commune ;

$OG \leftarrow SPO\_MergeGraph$ (O'_1, OC, O_2);

pour *N in* $\{NS\}$ **faire**
  $OG \leftarrow SPO\_AddEquivalentEntity$ (OG, $NS\{O_1\}$, $NS\{O_2\}$);
fin
pour *N in* $\{NIsa\}$ **faire**
  $OG \leftarrow SPO\_AddSubClass$ (OG, $NIsa\{O_1\}$, $NIsa\{O_2\}$);
fin

Algorithm 1: Algorithme de fusion d'ontologies.

Ainsi, le processus de fusion passe par quatre grandes étapes :
1) Apparier les deux ontologies. L'étape 1 vise à minimiser la différence entre les deux ontologies. Son rôle consiste à remplacer les entités de l'ontologie O_1 par leurs équivalents de l'ontologie O_2. Ce changement est réalisé par la règle de réécriture $RenameEntity$ (N_i, N_j) avec N_i est un nœud de O_1 et N_j est son équivalent dans O_2. Ce SPO est composé par :

– le graphe hôte qui est l'ontologie O_1 ;
– le LHS qui est le graphe constitué de l'ensemble des nœuds $\{N_i \in NE\}$;
– le RHS qui est le graphe formé par $\{N_j \in NE\}$.

2) Créer l'ontologie commune. L'étape 2 consiste à créer l'ontologie (OC) qui est le sous-graphe commun entre les deux ontologies. Elle est composée par les nœuds communs (NC) et les arêtes qu'ils partagent.
3) Créer l'ontologie globale. L'étape 3 permet de fusionner les ontologies avec la règle de réécriture $MergeGraph$. Cette règle de production a comme graphe hôte l'ontologie O'_1 (O_1 après modification). Son LHS est l'ontologie commune OC et son RHS est l'ontologie O_2. Elle a pour rôle l'intégration des deux ontologies et ceci en les liant par leurs entités communes. Le résultat de cette transformation donne une

première version de l'ontologie globale (OG) qui sera encore modifiée et enrichie par l'ajout des relations sémantiques et de subsomption.

4) Adapter l'ontologie globale. Étant donnée que la transformation de graphe nécessite la présence d'un *matcher* (morphisme m) entre le LHS et le graphe hôte, l'ajout des relations de subsomption et de relations sémantique doit être appliqué après la création de l'ontologie globale. Par conséquent, il faut appliquer les règles de réécriture $AddEquivalentEntity$ et $AddSubClass$ qui ont pour rôle l'enrichissement de l'ontologie globale sans affecter sa consistance. La vérification des inconsistances est assurée par les NAC. Ainsi, la règle de réécriture $AddEquivalentEntity$ ajoute un axiome d'équivalence entre deux entités (deux classes ou deux propriétés). A son tour, la règle de réécriture $AddSubClass$ ajoute l'axiome $subClassOf$ entre deux classes. Pour des raisons d'espace, que la règle $AddSubClass$ (C_1, C_2) est présentée (Figure 4). Elle est définie comme suit :

– NAC :

1) $C_1 \sqsubseteq C_2$, condition pour éviter la redondance ;

2) $C_2 \sqsubseteq C_1$, la relation de subsomption ne peut pas être symétrique ;

3) $C_1 \sqsubseteq \neg C_2$, les classes qui partagent une relation de subsomption ne peuvent pas être disjointes ;

4) $\exists C_i \in C(O) \cdot (C_1 \sqsubseteq C_i) \wedge (C_i \sqsubseteq C_2)$. S'il existe dans l'ontologie une classe C_i qui est à la fois la $subClass$ de la classe C_2 et la $superClass$ de C_1, alors, C_1 est déjà la $subClass$ de C_2 et il ne faut pas ajouter ce lien de subsomption ;

5) $\exists (C_i, C_j) \in C(O) \cdot (C_i \sqsubseteq C_1) \wedge (C_j \sqsubseteq C_2) \wedge C_i \sqsubseteq \neg C_j$, les classes qui partagent une relation de subsomption ne peuvent pas avoir des subClasses disjoints.

– LHS : $\{C_1, C_2\}$, les classes doivent exister dans l'ontologie.

– RHS : $(C_1 \sqsubseteq C_2)$, l'axiome doit être ajouté à l'ontologie.

– CHD : $\emptyset$.

Discussion. Deux types d'approche de fusion ont été distinguées dans la littérature (Raunich *et al.*, 2012) : 1) les approches symétriques qui permettent de fusionner deux ontologies tout en préservant la totalité de leur contenu dans l'ontologie globale (même les données redondantes) ; 2) les approches asymétriques qui ne gardent pas la totalité des concepts des ontologies.
Dans notre travail, nous présentons une approche asymétrique avec :

$$Merge(O_1, O_2) \neq Merge(O_2, O_1),$$
$$Merge(O_1, O_2) = Merge(O_2, Merge(O_1, O_2) \text{ et}$$
$$Merge(O_2, O_1) = Merge(O_1, Merge(O_2, O_1).$$

En effet, parmi les deux ontologies, nous considérons l'une comme source et l'autre comme destination. Les concepts de l'ontologie source seront préservés alors que seuls les concepts non redondants de l'ontologie destination, et qui n'affectent pas la consistance de l'ontologie source, seront ajoutés à l'ontologie globale.

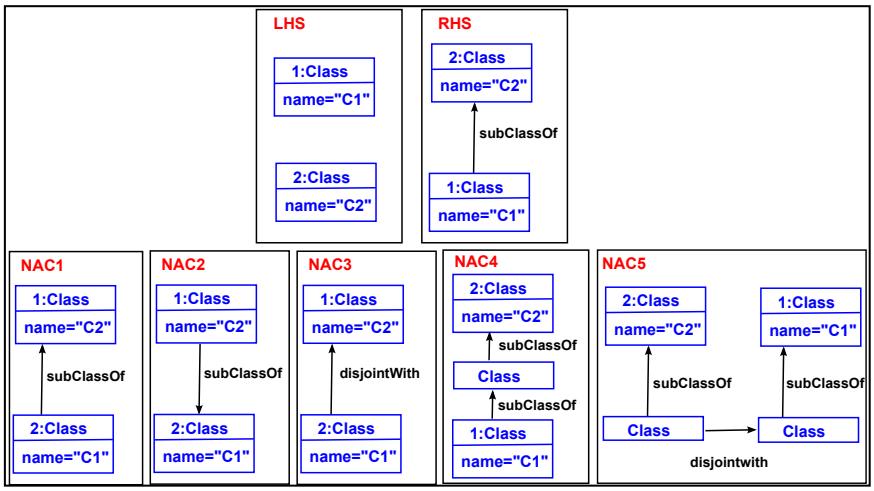

Figure 4. *Règle de réécriture du changement $AddSubClass$.*

5. Implémentation & évaluation

Pour valider notre proposition, l'approche a été implémentée et testée sur une dizaine d'ontologies dans différents domaines d'application. Ainsi, deux prototypes Java ont été développés :

– OntologiesMapping permet d'identifier la similarité syntaxique et sémantique entre ontologies. Il est basé sur l'ontologie linguistique WordNet(Miller, 1995) et la distance Levenshtein. Il prend en entrée deux ontologies et génère en sortie un fichier de mapping en XML. À noter que cet outil, dans son état actuel, ne gère pas les relations structurelles. Elles sont identifiées manuellement par un expert.

– GROM (Graph Rewriting for Ontology Merging) permet, d'une manière automatique, de fusionner deux ontologies selon le formalisme de grammaire de graphes typés. Il est basé sur l'API AGG (Attributed Graph Grammar)[3]ce qui a permet de réaliser les SPOs requis pour la fusion. L'outil prend en entrée deux ontologies au format GGX et leur fichier de mapping pour générer en sortie l'ontologie globale en format GGX. Les ontologies peuvent être facilement transformées du format GGX à OWL, et inversement, grâce à nos deux outils[4] GGXToOWL et OWLToGGX.

À noter que l'étape la plus coûteuse en temps et en ressources est la reconnaissance du graph pattern LHS à partir du graphe hôte G. Cette recherche est un problème NP-complet. Cependant, le temps de calcul reste tout à fait acceptable si la taille du graphe

3. `http://user.cs.tu-berlin.de/~gragra/agg`
4. Les codes sources sont disponibles en téléchargement, sous licence open source, via `http://mariem-mahfoudh.info/ksem2013/`

LHS est limitée. Dans la plupart des exemples de règles de réécriture, cette condition est satisfaite.

Le tableau 1 présente un ensemble d'ontologies fusionnées par l'outil GROM. Il précise leur nombre de concepts, leurs correspondances, le nombre de concepts du résultat de leur fusion et ainsi la valeur de couverture (i.e. le degré de préservation des concepts des ontologies source et destination dans le résultat de fusion).

Ontologies	**#Concepts**	**#Correspondances**	**#Fusion**	**Couverture**
Cars (Raunich *et al.*, 2012)	7 .. 6	6	13	1.00
CCAlps[5]	11 .. 8	4	16	0.84
Lebensmittel (Google, web) (Peukert *et al.*, 2010)	53 .. 59	15	112	1.00
Freizeit (dmoz, Google) (Peukert *et al.*, 2010)	71.. 67	67	71	0.51
Conference (ekaw, iasted)[6]	107 .. 182	10	280	0.97

Tableau 1: Le résultat de fusion de certaines ontologies avec l'outil GROM.

6. Travaux liés

Les travaux traitant la problématique de la fusion d'ontologies peuvent être classés en deux catégories : 1) les approches basées sur les technologies de web sémantique comme par exemple, IPrompt (Noy *et al.*, 2000), MMOMS (Li *et al.*, 2010), (Maree *et al.*, 2011), (Raunich *et al.*, 2014), etc. ; 2) les approches basées sur les spécifications algébriques et la théorie des catégories (Bouquet *et al.*, 2004; Zimmermann *et al.*, 2006; Cafezeiro *et al.*, 2007).

L'objectif de notre travail étant de proposer une approche formelle de fusion d'ontologies, nous avons principalement étudié les propositions existantes dans le domaine de spécifications algébriques. Ainsi, (Zimmermann *et al.*, 2006) ont présenté deux formalismes pour décrire l'alignement des ontologies : le "V-alignment" et le "W-alignment" qui se sont basés sur le concept "span" de la théorie des catégories. Ce travail présente une référence importante puisqu'il a défini les fondements de l'utilisation des catégories dans le domaine des ontologies. Cependant, la fusion n'est y

5. `http://www.ccalps.eu/`
6. `http://oaei.ontologymatching.org/2013/conference/`

traitée que comme une opération d'alignement. Par la suite, (Cafezeiro *et al.*, 2007) ont proposé l'utilisation des concepts de la théorie des catégories ("limit", "colimit" et "pushout") pour formaliser les opérations de composition et de fusion d'ontologies. Ce travail ne considère que les ontologies composées de classes et de propriétés. Il ne traite ni les individus ni les axiomes. Enfin, ces deux approches n'ont pas été implémentées et ne proposent qu'une formalisation. Dans notre travail, nous utilisons les approches algébriques et la théorie des catégories dans le cadre du formalisme des grammaires de graphes. Ceci nous a permis notamment d'implémenter l'approche (grâce aux outils proposés) et de profiter des conditions d'application (par exemple les NAC) pour éviter les inconsistances. Notre approche est également plus générale étant donné qu'elle traite les individus et les axiomes.

7. Conclusion et futurs travaux

Nous avons présenté dans cet article une nouvelle méthode de fusion d'ontologies basée sur la réécriture de graphes. L'idée étant de profiter de la puissance mathématique des grammaires de graphes et leurs approches algébriques afin de proposer un nouveau formalisme capable à la fois de représenter des ontologies et de suivre leur processus de fusion. Grâce à l'utilisation de SPO qui encapsule les détails complexes de la structure des ontologies en les considérant comme des objets d'une catégorie, la recherche des patterns et l'intégration des graphes semble une tâche facile pour l'utilisateur.

Comme perspectives à ce travail, nous envisageons une étude détaillée de différents conflits qui peuvent affecter le résultat de fusion. Nous visons également améliorer le résultat de l'alignement en intégrant des techniques structurelles.

8. Bibliographie

Barr M., Wells C., *Category theory for computing science*, vol. 10, Prentice Hall New York, 1990.

Bouquet P., Ehrig M., Euzenat J., Franconi E., Hitzler P., Krötzsch M., Serafini L., Stamou G., Sure Y., Tessaris S., Specification of a common framework for characterizing alignment, Knowledge Web Deliverable n° 2.2.1v2, University of Karlsruhe, 2004.

Cafezeiro I., Haeusler E. H., « Semantic interoperability via category theory », *26th international conference on Conceptual modeling (CM)*, Australian Computer Society, Inc., p. 197-202, 2007.

Châabane S., Jaziri W., Gargouri F., « OntoRoad : Ontologie spatiale pour le domaine routier », *Journées Francophones sur les ontologies (JFO)*, 2008.

Do H.-H., Rahm E., « COMA : a system for flexible combination of schema matching approaches », *28th international conference on Very Large Data Bases*, VLDB Endowment, p. 610-621, 2002.

Ehrig H., « Introduction to the algebraic theory of graph grammars (a survey) », *Graph-Grammars and Their Application to Computer Science and Biology*, Springer, p. 1-69, 1979.

Ehrig H., Heckel R., Korff M., Löwe M., Ribeiro L., Wagner A., Corradini A., « Algebraic Approaches to Graph Transformation-Part II : Single Pushout Approach and Comparison with Double Pushout Approach. », *in* G. Rozenberg (ed.), *Handbook of Graph Grammars*, p. 247-312, 1997.

Forestier G., Marion A., Benoit-Cattin H., Clarysse P., Friboulet D., Glatard T., Hugonnard P., Lartizien C., Liebgott H., Tabary J. et al., « Sharing object models for multi-modality medical image simulation : a semantic approach », *24th International Symposium onComputer-Based Medical Systems (CBMS)*, IEEE, p. 1-6, 2011.

Klein M., « Combining and relating ontologies : an analysis of problems and solutions », *IJCAI-2001 Workshop on ontologies and information sharing*, p. 53-62, 2001.

Levenshtein V. I., « Binary codes capable of correcting deletions, insertions and reversals », *Soviet physics doklady*, vol. 10, p. 707-710, 1966.

Li G., Luo Z., Shao J., « Multi-mapping based ontology merging system design », *2nd International Conference on Advanced Computer Control (ICACC)*, vol. 2, IEEE, p. 5-11, 2010.

Löwe M., « Algebraic approach to single-pushout graph transformation », *Theoretical Computer Science*, vol. 109, n° 1, p. 181-224, 1993.

Mahfoudh M., Forestier G., Thiry L., Hassenforder M., « Consistent Ontologies Evolution Using Graph Grammars », *Knowledge Science, Engineering and Management (KSEM)*, Springer, p. 64-75, 2013.

Mahfoudh M., Jaziri W., « Couplage entre BD et ontologies pour la satisfaction des requêtes SQL et SPARQL », *Journées Francophones sur les ontologies (JFO)*, p. 469-474, 2011.

Maree M., Alhashmi S. M., Belkhatir M., « A Unified Ontology Merging and Enrichment Framework », *23rd IEEE International Conference onTools with Artificial Intelligence (ICTAI)*, IEEE, p. 669-674, 2011.

Miller G. A., « WordNet : A Lexical Database for English », *Communications of the ACM*, vol. 38, n° 11, p. 39-41, 1995.

Noy N. F., Musen M. A., « Algorithm and tool for automated ontology merging and alignment », *17th National Conference on Artificial Intelligence (AAAI)*, AAAI Press / The MIT Press, p. 450-455, 2000.

Pavel S., Euzenat J., « Ontology Matching : State of the Art and Future Challenges », *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, n° 1, p. 158-176, 2013.

Peukert E., Massmann S., Koenig K., « Comparing Similarity Combination Methods for Schema Matching. », *GI Jahrestagung (1)*, Citeseer, p. 692-701, 2010.

Raunich S., Rahm E., « Towards a Benchmark for Ontology Merging », *On the Move to Meaningful Internet Systems : OTM 2012 Workshops*, Springer, p. 124-133, 2012.

Raunich S., Rahm E., « Target-driven merging of taxonomies with Atom », *Information Systems*, vol. 42, p. 1-14, 2014.

Rozenberg G., *Handbook of graph grammars and computing by graph transformation*, vol. 1, World Scientific, 1999.

Thiry L., Mahfoudh M., Hassenforder M., « A Functional Inference System for the Web », *International Journal of Web Applications*, vol. 6, p. 1-13, 2014.

Zimmermann A., Krotzsch M., Euzenat J., Hitzler P., « Formalizing ontology alignment and its operations with category theory », *Frontiers in Artificial Intelligence and Applications*, vol. 150, p. 277-288, 2006.