Approche formelle de fusion d'ontologies à l'aide des grammaires de graphes typés

Mariem Mahfoudh, Germain Forestier, Laurent Thiry, Michel Hassenforder

Approche formelle de fusion d'ontologies à l'aide des grammaires de graphes typés

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

*MIPS - EA 2332, Université de Haute-Alsace
12 rue des frères Lumière 68093 Mulhouse, France - prénom.nom@uha.fr

Résumé. L'article propose une approche formelle de fusion d'ontologies se reposant sur les grammaires de graphes typés. Elle se décompose en trois étapes : 1) la recherche de similarités entre concepts ; 2) la fusion des ontologies par l'approche algébrique SPO (Simple Push Out) ; 3) l'adaptation d'une ontologie globale par le biais de règles de réécriture de graphes. Contrairement aux solutions existantes, cette méthode offre une représentation formelle de la fusion d'ontologies ainsi qu'une implémentation fonctionnelle basée sur l'outil AGG.

1 Introduction

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* est généralement une tâche difficile et requiert un cadre formel capable de contrôler les différentes étapes de construction. Cet article propose ainsi de lever ce verrou scientifique par l'utilisation des grammaires de graphes typés (TGG) basées sur les approches algébriques. Les TGG sont un formalisme mathématique de représentation et de manipulation des graphes. Elles sont composées d'un graphe type (TG), un graphe hôte (G) et d'un ensemble de règles de production (P) appelées aussi règles de réécriture. Ces règles sont définies par une paire de graphes : 1) LHS (Left Hand Side) représente les pré-conditions de la règle ; 2) RHS (Right Hand Side) représente les post-conditions et doit remplacer LHS dans G. Les règles peuvent également avoir des conditions supplémentaires appelées NAC (Negative Application Conditions). La transformation de graphe consiste ainsi à définir comment un graphe G peut être transformé en un nouveau graphe G'. Pour cela, il doit exister un morphisme qui remplace LHS par RHS pour obtenir G'. Différentes approches ont été proposées pour appliquer ce remplacement. Dans ce travail, nous utilisons l'approche algébrique *Simple pushout SPO* (Löwe, 1993). Ainsi, appliquer une règle de réécriture à un graphe G, selon la méthode SPO, revient à : 1) trouver le LHS dans G ; 2) supprimer de G : $LHS - (LHS \cap RHS)$; 3) ajouter à G : $RHS - (LHS \cap RHS)$.

Dans notre travail (Mahfoudh et al., 2013), nous avons utilisé les TGG pour la formalisation et l'implémentation des changements ontologiques. Elles nous ont permis, grâce à leurs conditions d'application, d'assurer le contrôle du processus d'évolution et d'éviter les inconsistances. Dans cet article, nous nous basons sur le même formalisme pour définir une approche formelle de fusion des ontologies. Ce travail est financé par le projet Européen CCAlps (Creative Companies in Alpine space). Numéro de projet est 15-3-1-IT.

Peu de travaux ont étudié cette problématique avec les approches algébriques. Cafezeiro et Haeusler (2007) ont proposé une formalisation basée sur la théorie des catégories. Cependant, ils n'ont considéré que les ontologies composées de classes et de propriétés. Leur travail ne traite ni les individus ni les axiomes. Zimmermann et al. (2006) se sont essentiellement intéressés à l'alignement et la fusion d'ontologies n'est y traitée que comme une opération d'alignement. A noter que ces deux approches n'ont pas été implémentées. Dans ce travail, nous utilisons les approches algébriques dans le cadre des TGG. Ceci nous a permis notamment d'implémenter notre approche et de profiter des conditions d'application pour éviter les inconsistances.

2 Approche de fusion d'ontologies

Afin de mieux comprendre l'approche, nous nous servons des deux ontologies illustrées sur la Figure 1.

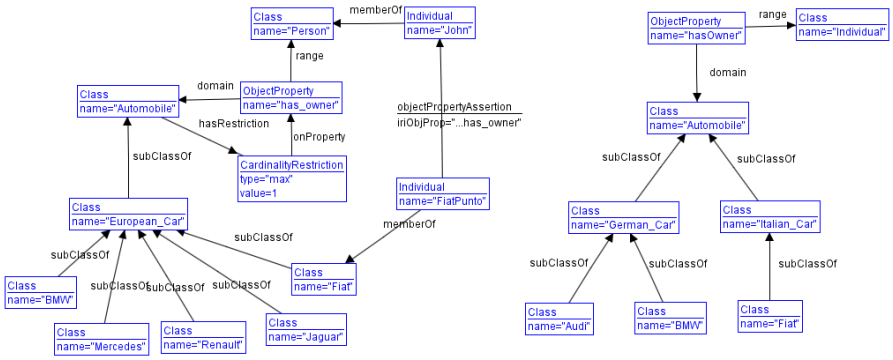

FIG. 1 – *Exemple des graphes hôtes : (gauche) Ontologie 1, (droite) Ontologie 2.*

2.1 Les ontologies et les grammaires de graphes typés

En adoptant le formalisme de grammaire de graphes typés sur les ontologies, on obtient :

- G est le graphe hôte qui représente une ontologie (voir l'exemple de la Figure 1).
- TG est le graphe type qui représente le méta-modèle de l'ontologie. Dans le cadre de cet article, c'est le méta-modèle d'OWL qui a été retenu. Ainsi, les types des nœuds sont les classes (C), individus (I), propriétés ($P = DP \cup OP$ avec DP sont les *datatype properties* et OP sont les *object properties*). Les types des arêtes sont les axiomes (A). Les restrictions (R) nécessitent à la fois les nœuds et les arêtes.
- P sont les règles de réécriture correspondantes aux changements ontologiques (ex. $AddClass$, $RemoveProperty$). Un changement ontologique $CH = (Name, NACs, LHS, RHS, CHDs)$ avec : 1) $Name$ précise le nom du changement ; 2) $NACs$ définissent les conditions qui ne doivent pas être satisfaites pour pouvoir appliquer le changement ontologique ; 3) LHS représente les pré-conditions du changement ; 4) RHS définit les post-conditions du changement ; 5) $CHDs$ sont les changements dérivés, c'est à dire

les règles de réécriture additionnelles qui sont attachées au CH pour corriger ses éventuelles inconsistances. Plus de détails et exemples des changements ontologiques sont présentés dans (Mahfoudh et al., 2013).

Nous décrivons dans ce qui suit notre approche de fusion des ontologies. A noter que l'approche a été implémentée en se basant sur l'API AGG (Algebraic Graph Grammar).

2.2 Recherche de similarité

Pour identifier la similarité entre les ontologies, nous nous sommes basés sur la distance de Levenshtein pour détecter les correspondances syntaxiques et l'ontologie linguistique WordNet pour identifier les correspondances sémantiques. Le processus de recherche de similarité prend ainsi deux ontologies et génère l'ensemble de correspondances suivantes : 1) CN est l'ensemble des nœuds communs ; 2) EN est l'ensemble des nœuds équivalents ; 3) SN est l'ensemble des nœuds partageant une relation sémantique. Les relations de subsomption ($IsaN$) sont identifiées manuellement. En considérant l'exemple des ontologies O_1 et O_2, on aura : 1) $CN = \{"Automobile", "Fiat, "BMW"\}$; 2) $EN = \{("has\_owner", "hasOwner")\}$; 3) $SN = \{("Person", "Individual")\}$. 4) $IsaN = \{("German\_Car", "European\_Car"), ("Italian\_Car", "European\_Car"), ("Mercedes", "German\_Car")\}$.

2.3 Fusion des ontologies

Le processus de fusion des ontologies avec l'approche SPO passe par trois étapes. La première vise à minimiser la différence entre les deux ontologies. Elle remplace ainsi les entités de l'ontologie O_1 par leurs équivalents de O_2 et ceci en appliquant la règle de réécriture $RenameNode$ (N_i, N_j) avec N_i est un nœud de O_1 et N_j est un nœud de O_2. On aura alors pour ce SPO : 1) le graphe hôte est l'ontologie O_1 ; 2) le LHS est le graphe constitué de l'ensemble des nœuds $\{N_i \in EN\}$; 3) le RHS est le graphe formé par $\{N_j \in EN\}$. L'étape 2 consiste à créer l'ontologie (CO) qui est le sous-graphe commun entre les deux ontologies. Elle est composée par les nœuds communs (CN) et les arêtes qu'ils partagent. La dernière étape fusionne les ontologies avec la règle de réécriture $MergeGraph(CO, O_2)$. Cette règle a comme graphe hôte l'ontologie O'_1 (O_1 après modification). Son LHS est l'ontologie commune CO et son RHS est l'ontologie O_2. Le résultat de cette transformation donne une première version de l'ontologie globale (OM) qui sera encore modifiée pour ajouter les relations sémantiques et de subsomption. Ainsi, pour notre exemple, la règle de réécriture $RenameObjectProperty$ est invoquée pour remplacer le nom de l'OP "has_owner" par "hasOwner". Après, l'ontologie commune (CO) sera créée et le SPO responsable d'intégrer les deux ontologies est exécuté pour générer l'ontologie (OM).

2.4 Adaptation de l'ontologie globale

Étant donné que la transformation de graphe nécessite la présence d'un *matcher* (morphisme m) entre LHS et le graphe hôte, l'ajout des relations de subsomption et sémantique doit être appliqué après la création de l'ontologie globale. Nous présentons dans cette section l'exemple de la règle de réécriture $AddSubClass$ qui ajoute l'axiome $subClassOf$ entre les nœuds de l'ensemble $IsaN$ (ex. $AddSubClass$ ("German_Car", "European_Car")). L'ajout

des axiomes est contrôlé par les $NACs$ pour ne pas altérer la consistance de l'ontologie. Ainsi la règle $AddSubClass$ (C_1, C_2) est définie par :

- NACs : 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 ce lien de subsomption ne doit pas être rajouté ; 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 : $\emptyset$.

3 Conclusion et futurs travaux

Nous avons présenté dans cet article une approche formelle de fusion d'ontologies basée sur les grammaires de graphes typés. L'approche proposée a été implémentée avec l'outil AGG qui permet de réaliser les SPOs requis pour la fusion. Comme perspectives de ce travail, nous envisageons l'étude de différents conflits qui peuvent affecter le résultat de fusion et la manière de les résoudre.

Références

Cafezeiro, I. et E. H. Haeusler (2007). Semantic interoperability via category theory. In *26th international conference on Conceptual modeling*, pp. 197–202.

Klein, M. (2001). Combining and relating ontologies : an analysis of problems and solutions. In *IJCAI-2001 Workshop on ontologies and information sharing*, pp. 53–62.

Löwe, M. (1993). Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science 109*(1), 181–224.

Mahfoudh, M., G. Forestier, L. Thiry, et M. Hassenforder (2013). Consistent ontologies evolution using graph grammars. In *Knowledge Science, Engineering and Management*, pp. 64–75. Springer.

Zimmermann, A., M. Krotzsch, J. Euzenat, et P. Hitzler (2006). Formalizing ontology alignment and its operations with category theory. *Frontiers in Artificial Intelligence and Applications 150*, 277–288.

Summary

This paper proposes a formal approach of merging ontologies based on typed graph grammars. It consists of three steps: 1) similarity search ; 2) merging ontologies by the algebraic approach Single PushOut ; 3) adaptation of a global ontology using the graphs rewriting rules. To validate our proposals, a prototype based on the AGG tool is implemented.