ONTOREV : un moteur de révision d'ontologies OWL

Thinh Dong, Chan Le Duc, Myriam Lamolle

LIASD - EA4383, IUT de Montreuil, Université Paris8, France
{dong, leduc, lamolle}@iut.univ-paris8.fr

English abstract We propose a Web-based application for revising an OWL ontology. The main functionality of this application consists in changing as less as possible an initial ontology when adding a new piece of knowledge such as an axiom, an assertion. To be able to ensure such a minimal change in preserving consistency of the resulting ontology, we use a set of models generated by a tableau algorithm to characterize the semantics of an ontology, and define a semantic distance between ontologies. Our Web-based application provides also other services such as visualizing an OWL ontology in a succinct syntax, inputting an OWL axiom/assertion.

Introduction. L'intelligence collective s'appuie sur des connaissances partagées. Afin que ces connaissances soient exploitables via des systèmes informatiques, elles doivent être représentées et manipulées en utilisant un mécanisme de raisonnement automatique. En outre, les connaissances qui reflètent la compréhension d'un domaine d'application évoluent au cours du temps. Ceci nécessite un mécanisme assurant à tout moment la cohérence sémantique de la totalité des connaissances du domaine représenté ; modifier une seule information modélisée implique de réviser/vérifier toute la sémantique portée par cette connaissance. Pour répondre à ces deux problématiques, nous proposons dans cet article une approche pour des modèles ontologiques consistant en un mécanisme de révision permettant la représentation et la gestion de l'évolution des connaissances.

Après une étude des approches existantes pour réviser des ontologies, nous avons proposé un nouvel algorithme tableau pour des ontologies d'expressivité $\mathcal{SHIQ}$ (incluse dans OWL-DL) qui garantit la priorité aux nouvelles connaissances, la cohérence de la nouvelle ontologie et les changements minimaux. Nous avons implémenté ce nouvel algorithme dans le prototype ONTOREV mis en ligne par des services Web afin de favoriser le travail collaboratif et la gestion de l'évolution de la modélisation collective d'une ontologie.

Fondement formel. Comme mentionné précédemment, les ontologies étudiées dans cet article sont exprimées en la logique de description $\mathcal{SHIQ}$ (Baader *et al.*, 2003) qui autorise l'expression de concepts, atomiques ou complexes, composés de constructeurs logiques booléens, de rôles (relations binaires) transitifs et inverses, de restrictions existentielle, universelle et numérique. Une ontologie consiste en un ensemble de connaissances dont chacune peut être soit une *assertion* de la forme $C(a)$ ou $R(a, b)$ où C est un concept, R un rôle, a, b des individus, soit un *axiome* de la forme $C \sqsubseteq D$ ou $R \sqsubseteq S$ où C, D sont des concepts et R, S des rôles. Comme toutes les logiques de description, la sémantique de $\mathcal{SHIQ}$ est fondée sur la théorie des modèles, *i.e.* elle utilise un domaine non-vide Δ et une fonction d'interprétation $\mathcal{I}$ pour associer à chaque

concept C un ensemble $C^{\mathcal{I}} \subseteq \Delta$, à chaque rôle R un ensemble $R^{\mathcal{I}} \subseteq \Delta \times \Delta$. Le couple $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ est un modèle de l'ontologie (*i.e.* elle est cohérente) si $\mathcal{I}$ vérifie toute contrainte sémantique (ou connaissance) contenue dans l'ontologie ; par exemple $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ pour tout axiome $C \sqsubseteq D$.

Soient $\mathcal{O}$ et $\mathcal{O}'$ deux ontologies en $\mathcal{SHIQ}$. La révision de $\mathcal{O}$ par $\mathcal{O}'$ consiste à construire une ontologie $\mathcal{O}^*$ conservant toutes les connaissances de $\mathcal{O}'$ et le plus possible les connaissances de $\mathcal{O}$. Nous avons adopté une approche fondée sur les modèles pour la construction de $\mathcal{O}^*$. Ceci veut dire que la sémantique d'une ontologie $\mathcal{O}$ est caractérisée par un ensemble fini de *modèles représentatifs*, noté $\mathsf{FM}(\mathcal{O})$, qui est construit par l'algorithme de tableau. Par conséquent, la révision de $\mathcal{O}$ par $\mathcal{O}'$ revient à construire un ensemble de modèles, noté $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$, qui inclut $\mathsf{FM}(\mathcal{O}')$ et la partie la plus grande de $\mathsf{FM}(\mathcal{O})$ compatible avec $\mathsf{FM}(\mathcal{O}')$. En particulier, si $\mathcal{O} \cup \mathcal{O}'$ est cohérente alors $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}') = \mathsf{FM}(\mathcal{O}) \cup \mathsf{FM}(\mathcal{O}')$, et $\mathcal{O}^* = \mathcal{O} \cup \mathcal{O}'$, *i.e.* la révision devient simplement l'ajout de $\mathcal{O}'$ dans $\mathcal{O}$. Les lecteurs intéressés peuvent trouver le détail de cet algorithme de révision dans l'article (Dong *et al.*, 2017).

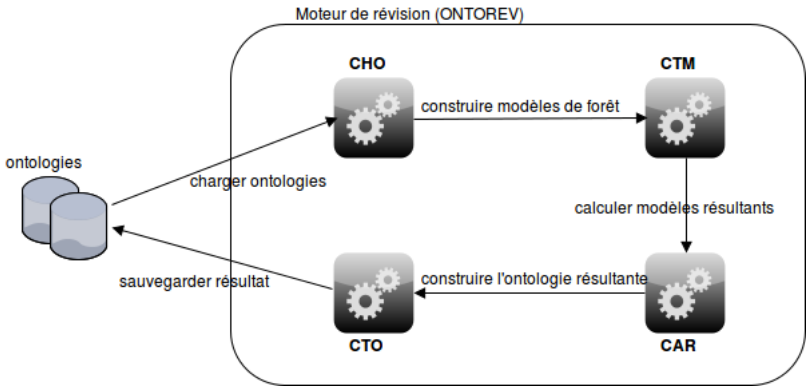

Fig. 1: Architecture d'ONTOREV

Architecture du noyau. L'algorithme de révision présenté dans la section précédente a été implémenté dans un prototype, appelé ONTOREV, et évalué avec plusieurs ontologies du monde réel. L'architecture d'ONTOREV est décrite dans la figure 1. Elle est composée de quatre modules principaux, à savoir, CHO qui charge une ontologie $\mathcal{O}$ à réviser et une autre ontologie $\mathcal{O}'$ contenant les assertions/axiomes à ajouter ; CTM qui construit les modèles de forêt de $\mathcal{O}$ et $\mathcal{O}'$; CAR qui calcule $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$ représentant les modèles de l'ontologie résultante à partir de $\mathsf{FM}(\mathcal{O})$ et $\mathsf{FM}(\mathcal{O}')$; enfin, CTO qui construit l'ontologie résultante $\mathcal{O}^*$ à partir de $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$ obtenu.

Le module CHO, qui dépend fortement de l'OWLAPI[1], doit effectuer également certains pré-traitements comme l'absorption, la suppression de tautologies, etc. Le module CTM implémente un algorithme tableau spécifique (Dong *et al.*, 2017) permettant de construire l'ensemble des modèles représentatifs $\mathsf{FM}(\mathcal{O})$ d'une ontologie $\mathcal{O}$.

[1] http://owlapi.sourceforge.net/

L'ensemble $\mathsf{FM}(\mathcal{O})$ peut être très grand s'il y a un nombre important de *non-déterminismes intrinsèques* (*e.g.* ceux qui ne proviennent pas directement du concept $\neg C \sqcup D$ pour un axiome $C \sqsubseteq D$). Le module CAR implémente l'algorithme calculant une distance sémantique entre deux modèles de forêt et plusieurs techniques d'optimisation pour réduire la cardinalité de $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$. Enfin, le module CTO génère l'ontologie résultante en $\mathcal{SHIQ}$ la plus "petite" sémantiquement qui prend pour modèles tous les élements dans $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$. Remarquons que l'existence d'une ontologie en $\mathcal{SHIQ}$ prenant *exactement* les éléments dans $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$ pour modèles n'est pas nécessaire.

Exemple. Pour illustrer le processus de révision, nous considérons une ontologie, notée $\mathcal{O}$, contenant les axiomes et assertions suivants :

(1) $\mathsf{Professor} \sqsubseteq \mathsf{Researcher} \sqcup \mathsf{Expert}$ (les professeurs sont des chercheurs ou experts),

(2) $\mathsf{Professor} \sqsubseteq \exists \mathsf{supervises.Student}$ (un professeur supervise au moins un étudiant),

(3) $\mathsf{Professor} \sqsubseteq (\geq 2\ \mathsf{teaches.Course})$ (un professeur enseigne au moins deux cours),

(4) $\mathsf{Professor}(\mathsf{Alex})$ (Alex est un professeur).

On souhaite maintenant réviser $\mathcal{O}$ en prenant en compte le fait que les chercheurs et experts ne supervisent aucun étudiant. On note $\mathcal{O}'$ une deuxième ontologie contenant deux nouveaux axiomes δ_1 et δ_2 qui expriment les nouvelles connaissances :

$\delta_1 : \mathsf{Researcher} \sqsubseteq \forall \mathsf{supervises.}(\neg \mathsf{Student})$ et $\delta_2 : \mathsf{Expert} \sqsubseteq \forall \mathsf{supervises.}(\neg \mathsf{Student})$

Si l'on ajoute $\mathcal{O}'$ dans $\mathcal{O}$ directement, l'union $\mathcal{O} \cup \mathcal{O}'$ sera incohérente car selon (4) Alex est un Professor et qui doit superviser un Student selon (2). Cependant, Alex doit être soit un Researcher, soit un Expert selon (1) ; ce qui contredit δ_1 ou δ_2. Ceci nécessite la révision de $\mathcal{O}$ afin que δ_1 et δ_2 soient prises en compte et les connaissances de $\mathcal{O}$ soient conservées le plus possibles. Pour ce faire, ONTOREV construit deux ensembles de modèles $\mathsf{FM}(\mathcal{O})$ et $\mathsf{FM}(\mathcal{O}')$. Ensuite, il extrait de $\mathsf{FM}(\mathcal{O}')$ l'ensemble de modèles, noté $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$, qui sont les plus proches (par rapport à la distance sémantique) des modèles dans $\mathsf{FM}(\mathcal{O})$. Enfin, ONTOREV génère à partir de $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$ l'ontologie résultante, notée $\mathcal{O}^*$, contenant les axiomes et assertion suivants :

$\mathsf{Expert} \sqsubseteq \forall \mathsf{supervises.}(\neg \mathsf{Student})$,

$\mathsf{Researcher} \sqsubseteq \forall \mathsf{supervises.}(\neg \mathsf{Student})$, $\mathsf{Professor}(\mathsf{Alex})$,

$\top \sqsubseteq (\mathsf{Professor} \sqcap \neg \mathsf{Researcher} \sqcap \mathsf{Expert} \sqcap \neg \mathsf{Course} \sqcap \neg \mathsf{Student} \sqcap \exists \mathsf{supervises.Student} \sqcap (\geq 2\ \mathsf{teaches.Course}) \sqcup (\mathsf{Professor} \sqcap \mathsf{Researcher} \sqcap \neg \mathsf{Expert} \sqcap \neg \mathsf{Course} \sqcap \neg \mathsf{Student} \sqcap \exists \mathsf{supervises.Student} \sqcap (\geq 2\ \mathsf{teaches.Course}) \sqcup (\mathsf{Student} \sqcap \forall \mathsf{supervises.}(\neg \mathsf{Student}) \sqcap (\leq 1\ \mathsf{teaches.Course}) \sqcap \neg \mathsf{Course} \sqcap \neg \mathsf{Professor} \sqcap \neg \mathsf{Researcher} \sqcap \neg \mathsf{Expert}) \sqcup (\mathsf{Course} \sqcap \forall \mathsf{supervises.}(\neg \mathsf{Student}) \sqcap (\leq 1\ \mathsf{teaches.Course}) \sqcap \neg \mathsf{Student} \sqcap \neg \mathsf{Professor} \sqcap \neg \mathsf{Researcher} \sqcap \neg \mathsf{Expert})$,

Remarquons que $\mathcal{O}^*$ peut inclure des axiomes de taille importante car ils proviennent des modèles trouvés dans $\mathsf{FM}(\mathcal{O} \oplus \mathcal{O}')$. Comme toutes les approches fondées sur les modèles, les axiomes de l'ontologie résultante construite par notre méthode décrivent les connaissances plutôt au niveau des modèles qu'au niveau de notre propre conception du domaine. Par conséquent, ils sont moins "parlant" que ceux des ontologies intiales. Pour évaluer ONTOREV, nous avons réalisé plusieurs tests avec différentes ontologies du monde réel. Les lecteurs intéressés peuvent trouver les résultats de ces tests dans l'article (Dong *et al.*, 2017).

Interface Web d'ONTOREV. Nous avons intégré ONTOREV dans une application Web[2] qui permet d'éditer, de réviser et d'interroger une ontologie OWL. Les avantages de cette interface sont que (i) tous les calculs lourds d'ONTOREV sont effectués sur un serveur puissant, (ii) l'affichage du contenu de l'ontologie de travail et la syntaxe de saisie d'un nouvel axiome/assertion sont facilités car exprimable en syntaxe Manchester (très utilisée par la communauté), (iii) les requêtes d'utilisateur sont analysées de façon centralisée. Cela permet d'optimiser la recherche des réponses en utilisant un mécanisme de cache ou d'apprentissage, et (iv) les requêtages courants sur une ontologie OWL tels que la vérification de la cohérence, la déduction d'un nouvel axiome/assertion y sont naturellement intégrés.

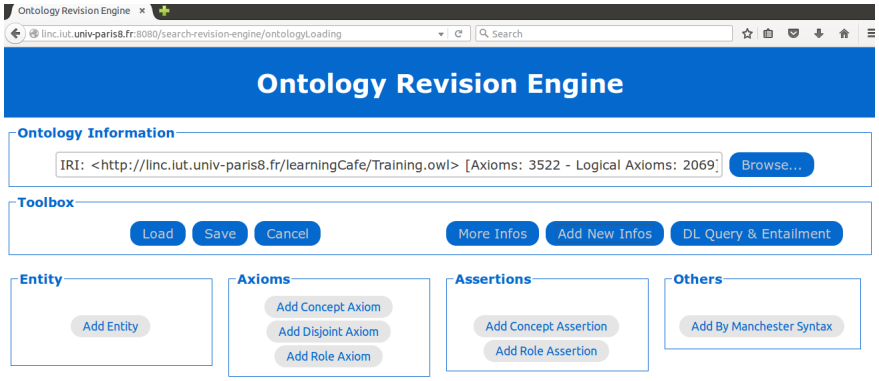

Fig. 2: Interface Web d'ONTOREV

Toutes les fonctionnalités accessibles via un navigateur Web sont aussi disponibles comme Web services du type REST. La page d'accueil (*cf. figure 2*) est découpée en deux zones :

(1) Ontology Information : Un utilisateur peut charger une ontologie en tapant l'URI de l'ontologie ou en choisissant (**Browse...**) une ontologie en local. Notons que si le bouton **Load** situé dans la zone **Toolbox** est cliqué mais que l'utilisateur n'a pas tapé de lien ni choisi une ontologie en local, l'ontologie *Training.owl* de la plate-forme *LearningCafé* se charge par défaut.

(2) Toolbox : contient les boutons permettant de charger une ontologie (**Load**), sauvegarder le résultat (**Save**), annuler le traitement (**Cancel**). De plus, l'utilisateur peut cliquer sur **More Infos** pour afficher les entités, les axiomes/assertions de l'ontologie de travail ; et **Add New Infos** lui permet de mettre à jour l'ontologie. Enfin, il peut aussi faire des requêtes sur l'ontologie en utilisant **DL Query & Entailment**. Cette fonctionnalité utilise les services du raisonneur HermiT (Shearer *et al.*, 2008) à l'heure actuelle.

Quand l'utilisateur clique sur **Add New Infos**, une nouvelle zone s'ouvre (*cf.* figure 2). Cette zone comporte plusieurs boutons **Entity, Axioms**, etc. permettant de saisir un

[2] http://linc.iut.univ-paris8.fr:8080/search-revision-engine/

nouvel axiome ou une nouvelle assertion à ajouter dans l'ontologie de travail. Ces boutons peuvent déclencher ONTOREV si l'axiome/assertion à ajouter n'est pas compatible avec ceux déjà existant dans l'ontologie de travail.

Conclusion et perspective. Nous avons présenté le prototype ONTOREV, implémenté en Java pour la révision d'ontologie OWL. Le service de révision d'ontologie basé sur ONTOREV a été intégré dans une interface Web permettant de faciliter l'accès au service et de profiter de la puissance de calcul d'un serveur. Comme perspective à court terme, nous visons à proposer une approche de révision hybride fondée non seulement sur la sémantique (modèles) mais aussi sur la syntaxe. Une telle approche améliorerait la lisibilité des axiomes de l'ontologie résultante.

References

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Dong *et al.*, 2017] Thinh Dong, Chan Le Duc, and Myriam Lamolle. Tableau-based revision for expressive description logics with individuals. *J. Web Sem.*, 45:63–79, 2017.

[Shearer *et al.*, 2008] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In Alan Ruttenberg, Ulrile Sattler, and Cathy Dolbear, editors, *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, Karlsruhe, Germany, October 26–27 2008.