Moteur de révision d'ontologie en $\mathcal{SHIQ}$

Thinh Dong, Myriam Lamolle, Chan Le Duc, Philippe Bonnot

LIASD (EA-4383)
Université Paris 8 - IUT de Montreuil, France
dong, lamolle, leduc, bonnot@iut.univ-paris8.fr

RÉSUMÉ. *L'intelligence collective s'appuie de plus en plus sur des représentations ontologiques de la connaissance. Cependant modifier une information implique de réviser/vérifier toute la sémantique portée par cette connaissance. Après une étude sur les approches existantes pour réviser des ontologies, nous avons proposé un nouvel algorithme tableau pour des ontologies d'expressivité $\mathcal{SHIQ}$ 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 *et mis en ligne par des services Web afin de favoriser le travail collaboratif et la prise de décision de groupe lors de l'accomplissement d'une tâche.*

ABSTRACT. *The more and more collective intelligence benefits from ontological knowledge representations. Developing collaboratively an ontology would need to often revise it. However, changing a portion of represented knowledge of an ontology may lead to change the semantics of the whole ontology. We have proposed a novel tableau algorithm for revising an ontology expressed in the $\mathcal{SHIQ}$ description logic following a deep investigation of existing approaches to ontology revision. This algorithm ensures integration of the new knowledge into the ontology, consistency of the resulting ontology and minimal changes. We have implemented this algorithm and integrated it within a web-based prototype, called* OntoRev. *This prototype provides access to functions related to ontology revision via web services and supports to develop and maintain an ontology in a collaborative way.*

MOTS-CLÉS : *intelligence collective, ontologie, révision, raisonnement, services Web.*

KEYWORDS: *collective intelligence, ontology, revision, reasoning, Web services.*

DOI:10.3166/ISI.23.2.39-59.

1. Introduction

Le monde numérique d'aujourd'hui qui manipule des millions de données par jour est censé favoriser ou faciliter nos prises de décisions car cet état de fait suggère que toutes ces données sont notre savoir commun. Par voie de conséquence, cela suggère aussi que la fusion de notre savoir et de nos connaissances par le biais du Web représente l'intelligence collective disponible. Or, il a été démontré que l'intelligence collective d'un groupe n'était en aucune façon la somme des intelligences individuelles de ce groupe (Lévy, 1994) et que cela ne suffit pas pour améliorer les prises de décisions, l'aboutissement d'un projet, la vie du groupe, etc. (Hansen, Vaagen, 2016).

Il semble donc que pour pouvoir appréhender cette intelligence collective ayant comme fondement les connaissances sous-jacentes d'un groupe collaborant ou coopérant, il est nécessaire de disposer de modèles de représentation évolués, porteurs de sens en soi, facilitant l'interopérabilité entre systèmes (communication entre machines et/ou entre machines et humains) afin d'enrichir le noyau de connaissance du groupe mais aussi de vérifier en amont qu'il n'y ait pas de contradiction entre les connaissances de chaque membre et/ou des connaissances issues de l'enrichissement. De nouveaux modèles dits ontologiques ont fait une percée remarquée ces dernières décennies dans ce but (Maleszka, Nguyen, 2015), (Krötzsch, Vrandecic, 2011), (Trappey *et al*., 2014). Dans l'ingénierie des connaissances, une ontologie est une spécification formelle et explicite d'une conceptualisation partagée d'un domaine. L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, qui peut être réel ou imaginaire. Les ontologies ont été appliquées dans un large éventail de domaines pratiques tels que l'e-Science, l'e-Commerce, l'informatique médicale, la bio-informatique, le Web sémantique, etc.

Les applications fondées sur la sémantique encapsulent souvent un ensemble d'ontologies qui représentent les connaissances impliquées dans différentes sources de données. Certaines de ces ontologies évoluent constamment car, non seulement les données sont mises à jour, mais aussi l'environnement des applications et les besoins des utilisateurs sont modifiés au cours du temps. Pour gérer les changements de ces ontologies, nous avons étudié les problématiques liées à la révision d'ontologies afin de les garder cohérente en minimisant la perte éventuelle de sémantique. Parmi les premiers travaux portant sur la révision d'ontologies, le cadre de travail *AGM* (Alchourrón, Gärdenfors et Makinson (Alchourrón *et al*., 1985)) a fourni un ensemble de contraintes logiques à respecter lors de la définition des opérations de révision. Ces contraintes portent sur la priorité de nouvelles connaissances, la cohérence de connaissances et la minimalité des changements. Plusieurs recherches ont proposé des opérations de révision satisfaisant les postulats *AGM* (Alchourrón *et al*., 1985). Cependant, ces opérations de révision peuvent engendrer une ontologie qui n'est plus exprimable dans la logique de l'ontologie initiale. De plus, les ontologies utilisées sont représentées dans les logiques qui ne sont pas assez expressives, *e.g.* DL-Lite.

L'objectif de ce travail est de faire une étude approfondie des possibilités d'extension ou de reformulation des opérateurs de révision existants respectant les postulats

AGM pour des ontologies d'expressivité $\mathcal{SHIQ}$ et de proposer de nouveaux algorithmes palliant les inconvénients inhérents à ces opérateurs mais aussi des solutions pratiques permettant d'implémenter un prototype qui valide les résultats théoriques.

Nous allons donc dans un premier temps rappeler brièvement le contexte de cette étude dans la section 2 qui est nécessaire pour comprendre les principes de la révision d'ontologie présentés dans la section 3. Puis, dans la section 4, nous allons détailler notre technique de révision pour des ontologies d'expressivité SHIQ. Le moteur de révision *OntoRev* sera détaillé dans la section 5 et son application en ligne dans la section 6. Enfin, nous ferons le bilan de ce premier prototype et indiquerons les perspectives possibles.

2. Contexte de l'étude

Dans cette section, nous allons délimiter le contexte général de cette étude en présentant tout d'abord les fondements des logiques de description ; et, plus précisément, la logique expressive $\mathcal{SHIQ}$ pour laquelle nous essayons de solutionner la problématique de la révision d'ontologies.

2.1. Logiques de description

Les **logiques de description** (LD) sont une famille de langages de représentation de connaissances qui peuvent être utilisés pour représenter la connaissance terminologique d'un domaine d'application d'une manière formelle et structurée (Baader, Nutt, 2003). Ces langages ont été introduits dans les années 80, et leur développement fut fortement influencé par les travaux sur la logique des prédicats, les schémas (frames) (Minsky, 1981) et les réseaux sémantiques (Fournier-Viger, 2005).

Les LD utilisent les notions de *concept*, *rôle* et *individu*. Un concept C correspond à une « classe d'éléments », les rôles R aux « liens entre les éléments » et les individus, quant à eux, aux éléments d'un univers donné ; mais aussi un ensemble de constructeurs tels que la conjonction ($\sqcap$), disjonction ($\sqcup$), restriction universelle ($\forall R.C$), restriction existentielle ($\exists R.C$), restrictions de cardinalité supérieure et inférieure ($\geq n.R$, $\leq n.R$) et négation primitive (ou complète), etc.

La sémantique des logiques de description est spécifiée par des interprétations. Une interprétation $\mathcal{I}$ est une paire $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$, où $\Delta^{\mathcal{I}}$ est un ensemble non vide appelé *domaine* et $.^{\mathcal{I}}$ est une fonction d'interprétation qui associe chaque concept atomique A à un sous-ensemble $A^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$, chaque rôle atomique P à une relation binaire $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, et chaque nom d'individu a à un élément $a^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$.

La logique $\mathcal{SHIQ}$ est fondée sur une extension de la bien connue LD $\mathcal{ALC}$ en incluant les rôles transitifs primitifs (Sattler, 1996). Rappelons que la logique $\mathcal{ALC}$ augmentée par les rôles transitifs est notée $\mathcal{S}$ (Horrocks *et al.*, 1999). Cette LD de base est ensuite étendue avec la hiérarchie de rôles ($\mathcal{H}$), les rôles inverses ($\mathcal{I}$) et les restrictions de cardinalité qualifiées ($\mathcal{Q}$).

2.2. Raisonnement et algorithme de tableau

Une fois la base de connaissance représentée par une LD plus ou moins expressive, il s'agit pour tout raisonneur d'offrir deux services à savoir la vérification de la cohérence (pas de contradiction terminologique et/ou assertive) et l'inférence de connaissance. Pour ce faire, l'algorithme de tableau (Baader, Sattler, 2001) construit un arbre qui développe des branches jusqu'à trouver un modèle cohérent. Si ce n'est pas le cas, le raisonneur en déduit que l'ontologie est incohérente.

Ce genre d'algorithme est fondé sur deux structures à savoir la structure « tableau » elle-même et le graphe de complétion. Brièvement, la structure de tableau d'un concept représente un modèle qui peut être infini. Cela traduit la satisfiabilité de tout concept et axiomes d'inclusion de rôles en la satisfiabilité des contraintes sémantiques imposées localement à chaque individu du tableau. Cette caractéristique des tableaux est appelée la propriété de satisfiabilité locale. Un graphe de complétion, quant à lui, est une représentation finie à partir de laquelle un tableau peut être construit. Pour vérifier la satisfiabilité d'un concept, les algorithmes de tableau essaient de construire un graphe de complétion dont la terminaison est assurée par une technique dite de blocage. Elle fournit une condition de terminaison et garantit la correction et la complétude. L'idée sous-jacente du mécanisme de blocage est de détecter des « boucles » qui sont des parties se répétant d'un graphe de complétion

Lors de l'application de l'algorithme tableau sur des ontologies exprimables dans la logique expressive $\mathcal{SHIQ}$, Horrocks *et al.* (Horrocks *et al.*, 1999) ont proposé un ensemble de règles et de techniques pour pouvoir traiter la transitivité de rôles, la hiérarchie de rôles, les rôles inverses et les restrictions de cardinalité qualifiées. Dans le pire des cas, la complexité du raisonnement de l'algorithme de tableau pour $\mathcal{SHIQ}$ est EXPTIME-complet (Tobies, 2001). Cependant, en pratique, le comportement des algorithmes de tableau est souvent acceptable (Baader *et al.*, 2003). Notons que la technique de blocage nous permet de prolonger les chemins par les « boucles » pour satisfaire les contraintes sémantiques imposées par les rôles transitifs quand ces derniers sont ajoutés aux bases de connaissances que sont les ontologies.

3. Principes de la révision d'ontologie

Le problème de la révision d'ontologies exprimables en OWL-DL (appelée ontologies en LD) est étroitement lié au problème de la révision de croyances qui a été largement discuté dans la littérature. L'approche la plus connue dans la révision de croyances est le paradigme AGM d'Alchourrón, Gärdenfors et Makinson (Alchourrón *et al.*, 1985).

Katsuno et Mendelzon (Katsuno, Mendelzon, 1991) ont adapté les postulats AGM dans le cadre de la logique propositionnelle comme suit, où $\omega \circ \mu$ désigne le résultat de la révision d'une formule ω par une nouvelle formule μ :

Ro1 $\omega \circ \mu$ implique μ,

Ro2 si $\omega \wedge \mu$ est satisfiable alors $\omega \circ \mu \equiv \omega \wedge \mu$,

Ro3 si μ est satisfiable alors $\omega \circ \mu$ est aussi satisfiable,

Ro4 si $\omega_1 \equiv \omega_2$ et $\mu_1 \equiv \mu_2$, alors $\omega_1 \circ \mu_1 = \omega_2 \circ \mu_2$,

Ro5 $(\omega \circ \mu_1) \wedge \mu_2$ implique $\omega \circ (\mu_1 \wedge \mu_2)$;

Ro6 Si $(\omega \circ \mu_1) \wedge \mu_2$ est satisfiable, alors $\omega \circ (\mu_1 \wedge \mu_2)$ implique $(\omega \circ \mu_1) \wedge \mu_2$.

En pratique, les approches classiques de révision lors de la construction d'un opérateur de révision peuvent être classées en deux grandes catégories à savoir les approches fondées sur les *syntaxes (approches syntaxiques)* et celles fondées sur les *modèles (approches sémantiques)* (Eiter, Gottlob, 1992). Les approches syntaxiques manipulent directement des entités syntaxiques telles que les formules d'une base de connaissances. En ce sens, elles nous ont paru moins riches que les approches sémantiques.

Les principales problématiques des approches sémantiques sont liées à la définition d'une distance entre les modèles et à la façon de calculer le résultat de la révision à partir des modèles sélectionnés en fonction de la distance définie. Des opérateurs spécifiques de révision (Dalal, 1988), (Satoh, 1988) ont été proposés sur la base de la distance entre des modèles de la base de connaissances et de nouvelles connaissances. L'opérateur de révision de Dalal (Dalal, 1988) utilise la cardinalité pour mesurer la distance alors que celui de Satoh (Satoh, 1988) utilise l'ensemble d'inclusion (*containment set* en anglais) fondé sur la différence symétrique $(S \setminus S') \cup (S' \setminus S)$ pour deux ensembles S, S'.

Lors de l'adaptation des approches sémantiques à la révision d'une ontologie en LD, il existe aussi des problèmes qui peuvent survenir pendant le traitement des modèles d'ontologies en LD. Premièrement, les ontologies en LD peuvent avoir un nombre infini de modèles qui rendent impossible la construction d'une ontologie résultant d'une révision à partir des modèles. Deuxièmement, les modèles d'une ontologie en LD ont généralement des structures complexes (éventuellement infinies), ce qui peut nécessiter une définition complexe de la distance entre deux modèles. Troisièmement, il peut ne pas exister une unique ontologie qui admette exactement un ensemble donné de modèles (Grau *et al*., 2012), (Kharlamov *et al*., 2013).

En dépit de ces problèmes, il y a eu de nombreuses tentatives pour adapter les approches sémantiques de révision aux ontologies en LD (citons (Qi, Du, 2009), (Wang *et al*., 2010)).

Un résultat qui établit des relations entre des approches fondées sur les syntaxes et celles fondées sur les modèles a été présenté par Qi et ses collègues (Qi *et al*., 2015). Ils ont prouvé qu'un opérateur de révision reposant sur la syntaxe peut être utilisé pour approximer les opérateurs de révision fondés sur un modèle dans DL-Lite$_{\mathcal{R}}$ proposés par Kharlamov et ses collègues (Kharlamov *et al*., 2013). En d'autres termes, le résultat de l'opérateur de révision fondé sur les syntaxes est une « meilleure » approximation de deux opérateurs de révision basés sur les modèles qui n'ont pas encore eu de procédure de calcul. Les auteurs (Qi *et al*., 2015) ont également proposé un algorithme basé sur les graphes pour calculer la révision d'ontologies en utilisant des techniques

de base de graphes pour calculer la révision d'ontologies. Dans leur graphe, chaque nœud représente un concept basique ou un rôle basique à partir de la signature d'une ontologie DL-Lite, et chaque arc représente une assertion dans l'ontologie.

4. Notre approche de révision pour des ontologies en $\mathcal{SHIQ}$

4.1. Révision par arbre de complétion

Pour résoudre les problèmes liés à la révision d'ontologies, dans notre approche sémantique, nous utilisons un ensemble fini de structures finies, à savoir un *arbre de complétion*, pour représenter un ensemble de modèles éventuellement infini d'une ontologie en $\mathcal{SHIQ}$. Ces structures finies qui ont un pré-ordre total permettent de déterminer la différence sémantique entre deux ontologies représentées en tant que deux ensembles de modèles. En effet, la révision d'une ontologie $\mathcal{O}$ par une autre ontologie $\mathcal{O}'$ peut être réduite à la sélection des modèles « appropriés » à partir de l'ensemble de tous les modèles admis par $\mathcal{O}'$ tels que les modèles sélectionnés soient les plus proches des modèles de $\mathcal{O}$. Les modèles sélectionnés sont ensuite utilisés pour construire une ontologie résultante de la révision de $\mathcal{O}$ par $\mathcal{O}'$.

Pour illustrer cette idée, considérons l'exemple 1.

EXEMPLE 1. — *Soit $\mathcal{O}$ une ontologie contenant les axiomes :*

$$\top \sqsubseteq \mathsf{BakerVideo} \sqcup \mathsf{PastryVideo} \sqcup \mathsf{User} \quad (1)$$

$$\mathsf{BakerVideo} \sqsubseteq \neg\mathsf{PastryVideo} \quad (2)$$

$$\mathsf{User} \sqsubseteq \neg\mathsf{BakerVideo} \sqcup \neg\mathsf{PastryVideo} \quad (3)$$

$$\mathsf{User} \sqsubseteq \exists\mathsf{hasSeen}.(\mathsf{BakerVideo} \sqcup \mathsf{PastryVideo}) \quad (4)$$

L'axiome 1 dit que « tout est une instance de la classe BakerVideo *ou* PastryVideo *ou* User *». Les concepts* BakerVideo *et* PastryVideo *sont disjoints (cf. axiome 2), de même* User *et chaque* BakerVideo *et* PastryVideo *(cf. axiome 3). L'axiome 4 dit qu' « une instance de* User *doit avoir vu (*hasSeen*) une vidéo de* BakerVideo *ou de* PastryVideo *». Une meilleure compréhension du domaine de la boulangerie-pâtisserie peut nous conduire à ajouter l'axiome suivant provenant d'une autre ontologie $\mathcal{O}'$:*

$$\top \sqsubseteq (\mathsf{BakerVideo} \sqcap \mathsf{PastryVideo}) \sqcup \mathsf{User} \quad (5)$$

Ce nouvel axiome stipule que « tout est une instance soit de BakerVideo *et* PastryVideo*, soit de* User *» (cf. axiome 5). L'ontologie $\mathcal{O} \cup \mathcal{O}'$ est incohérente car l'axiome 5 contredit l'axiome 2. Rappelons que notre but est de construire une nouvelle ontologie $\mathcal{O}^*$ qui soit « compatible » avec les axiomes provenant de $\mathcal{O}'$ telle que $\mathcal{O}^*$ soit sémantiquement la plus proche de $\mathcal{O}$ (i.e. correspondant à un changement minimal). Nous pouvons vérifier que les deux arbres de complétion T_1 et T_2 de la figure 1 produisent des modèles de $\mathcal{O}$.*

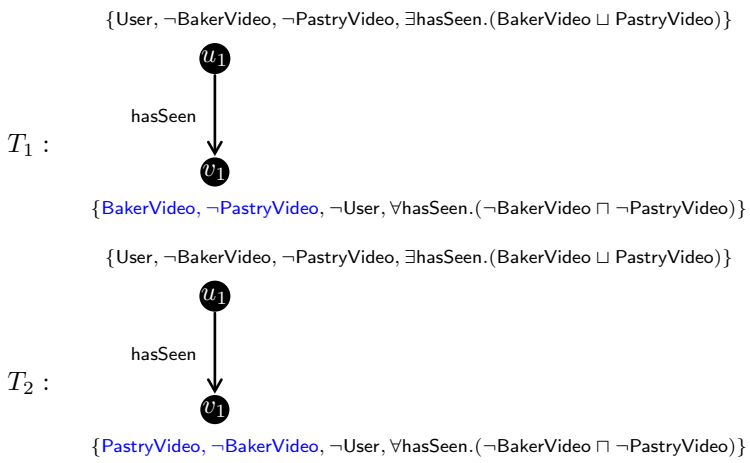

Figure 1. Exemple d'arbres de complétion pour l'ontologie $\mathcal{O}$

De même, nous pouvons vérifier que les deux arbres T'_1, T'_2 de la figure 2 produisent des modèles de $\mathcal{O}'$.

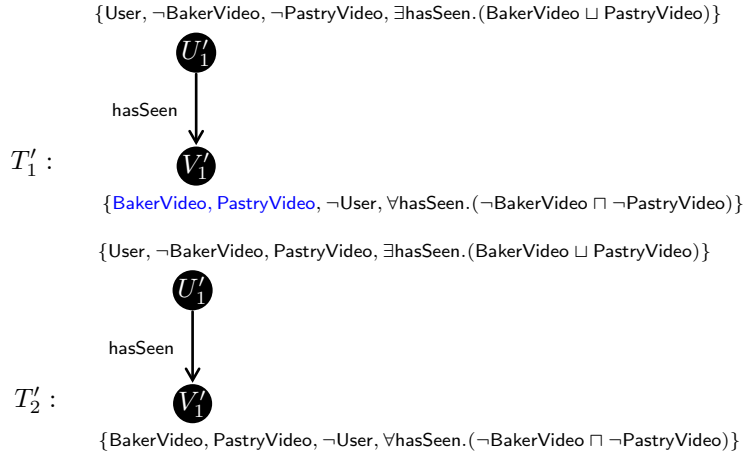

Figure 2. Exemple d'arbres de complétion pour l'ontologie $\mathcal{O}'$

En définissant une distance entre des arbres de complétion fondée sur la similarité structurelle, il est plausible de dire que T'_1 est plus proche de T_1 et T_2 que T'_2. Donc, une ontologie résultante $\mathcal{O}^*$ devrait admettre T'_1 plutôt que T'_2.

Une autre problématique à résoudre dans notre approche est la possibilité de la non existence d'une ontologie de révision telle qu'elle puisse être exprimée dans la logique

utilisée pour l'expression des ontologies initiales $\mathcal{O}, \mathcal{O}'$, et qu'elle admette *exactement* un ensemble d'arbres de complétion comme ses modèles. Pour cette raison, nous empruntons des travaux de De Giacomo et ses collègues la notion d'*approximation maximale* (De Giacomo *et al.*, 2007) nous permettant de générer une ontologie sémantiquement minimale d'une révision telle qu'elle admette un ensemble de modèles construit à partir de $\mathcal{O}$ et $\mathcal{O}'$.

Ceci nous amène à proposer une méthode permettant de caractériser la sémantique d'une ontologie en $\mathcal{SHIQ}$ en utilisant un ensemble fini de *graphes de complétion* plutôt que des arbres de complétion.

4.2. Révision par graphe de complétion

Nous proposons une extension de l'approche fondée sur les modèles présentée dans la sous-section précédente pour réviser des ontologies en $\mathcal{SHIQ}$ avec individus. La construction de notre procédure de révision est fondée sur les points suivants : (i) utiliser des graphes de complétion générés par un nouvel algorithme pour caractériser la sémantique d'une ontologie en $\mathcal{SHIQ}$. Cet algorithme doit construire un ensemble de graphes de complétion, noté $\mathsf{FM}(\mathcal{O})$, pour une ontologie $\mathcal{O}$ en considérant tous les cas non-déterministes intrinsèques au lieu de construire un seul graphe de complétion comme ce que les algorithmes de tableau existants font ; (ii) définir une distance sur un ensemble de graphes de complétion pour aborder le principe du changement minimal. Étant donné une ontologie $\mathcal{O}'$ contenant de nouveaux axiomes devant être pris en compte lors de la révision, cette distance peut aider à choisir des graphes de complétion à partir de $\mathsf{FM}(\mathcal{O}')$ de telle façon qu'ils soient sémantiquement les plus proches de ceux de $\mathsf{FM}(\mathcal{O})$. Une ontologie de révision de $\mathcal{O}$ par $\mathcal{O}'$ doit admettre les graphes de complétion choisis comme modèles ; (iii) introduire la notion d'approximation d'ontologie pour surmonter le problème de l'inexprimabilité. Notre procédure de révision renvoie une approximation d'ontologie exprimable en $\mathcal{SHIQ}$ et la plus petite en terme de sémantique.

Pour illustrer l'idée sous-jacente à cette construction, nous considérons l'exemple 2.

EXEMPLE 2. — *Étant donnée une ontologie* UNI *dans le tableau suivant :*

α_1 : $\mathsf{Professor} \sqsubseteq \mathsf{Researcher} \sqcup \mathsf{Expert}$	Un professeur est un chercheur ou un expert
α_2 : $\mathsf{Professor} \sqsubseteq \exists \mathsf{supervises.Student}$	Un professeur supervise au moins un étudiant
α_3 : $\mathsf{Professor} \sqsubseteq (\geq 2\ \mathsf{teaches.Course})$	Un professeur enseigne au moins deux cours
β : $\mathsf{Professor(Alex)}$	Alex est un professeur

Tableau 1. Ontologie UNI

Supposons que les chercheurs et les experts ne supervisent aucun étudiant, nous devons alors ajouter dans UNI *cette connaissance formulée par les axiomes suivants :*

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

Cependant, la présence de δ_1 *et* δ_2 *rendra* UNI *incohérente, et cela nécessite une révision pour maintenir la cohérence d'*UNI*. Une des façons d'appliquer la révision est de supprimer certains axiomes d'*UNI*. Intuitivement, on peut éliminer soit* α_1 *soit* α_2 *pour maintenir sa cohérence. Dans le cas où* α_1 *est retiré alors l'ontologie obtenue* $\widehat{\mathcal{O}} = \{\alpha_2, \alpha_3, \beta, \delta_1, \delta_2\}$ *est cohérente. Cependant, la connaissance « Un professeur est un expert » dans* α_1 *ne contredit pas l'ontologie* $\widehat{\mathcal{O}}$ *mais elle a été enlevée en même temps qu'*α_1*. En d'autres termes, l'objectif est de construire une nouvelle ontologie* $\mathcal{O}^*$ *« compatible » avec les axiomes d'*UNI *telle que* $\mathcal{O}^*$ *est sémantiquement la plus proche possible d'*UNI*; ce qui signifie un changement minimal. Nous pouvons vérifier que les forêts de complétion* $\mathcal{F}_1$*,* $\mathcal{F}_2$ *de la figure 3 donnent des modèles d'*UNI*. Dans ce cas, chaque noeud étiqueté représente un individu et chaque arc représente une relation entre deux individus.*

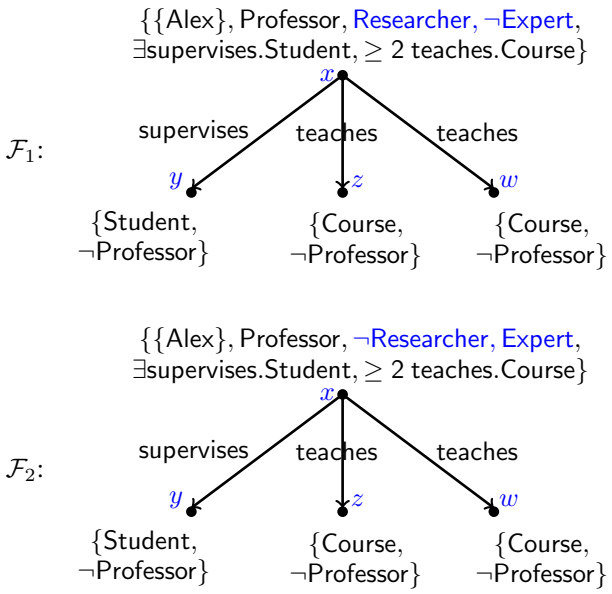

*Figure 3. Forêts de complétion donnant des modèles d'*UNI

De même, nous pouvons vérifier que les deux forêts de la figure 4 donnent des modèles de $\{\delta_1, \delta_2\}$.

Si nous définissons une distance entre les forêts de complétion en fonction de la similarité structurelle, il serait plausible de dire que $\mathcal{F}_1'$ *est plus proche de* $\mathcal{F}_1$ *et* $\mathcal{F}_2$ *que* $\mathcal{F}_2'$*. Donc, une ontologie de révision* $\mathcal{O}^*$ *devrait admettre* $\mathcal{F}_1'$ *plutôt que* $\mathcal{F}_2'$*.*

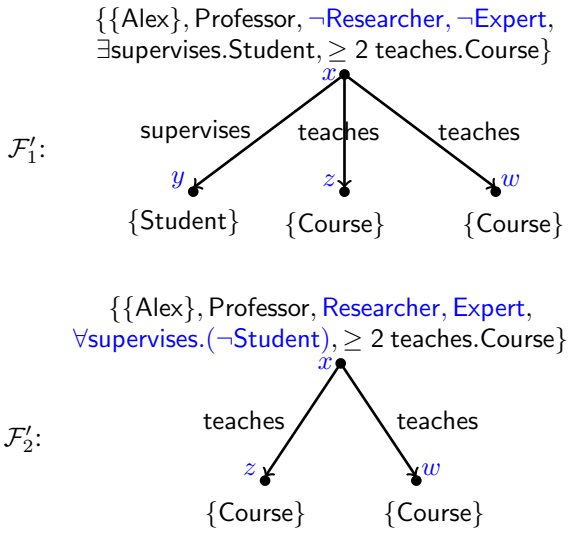

Figure 4. Forêts de complétion donnant des modèles de $\{\delta_1, \delta_2\}$

5. Moteur de révision *OntoRev*

Nous avons mis en place un moteur de révision en tant que prototype, appelé OntoRev. De même que les raisonneurs de la logique de description tels qu'HermiT (Shearer *et al.*, 2008), Pellet (Sirin *et al.*, 2007), FaCT++ (Tsarkov, Horrocks, 2006), nous avons implémenté dans OntoRev diverses techniques d'optimisation telles que l'absorption et les blocages pour réduire les cas non déterministes résultant de la disjonction de concepts. Nous avons également appliqué la technique de blocage de noyau (*core blocking* en anglais) (Glimm *et al.*, 2010) à côté de la technique de blocage de paire (*pairwise blocking* en anglais) pour réduire la taille des forêts de complétion. Contrairement aux raisonneurs de tableau existants, nous devons explorer tous les cas intrinsèques non déterministes impliqués dans les ontologies pour construire toutes les forêts de complétion. Dans la version actuelle d'OntoRev, certaines techniques d'optimisation telles que l'élagage des points de retour (« pruning of backtracking points ») pour traiter le non-déterminisme intrinsèque n'ont pas été mises en oeuvre. Notons que le manque d'implémentation d'optimisations avancées peut ralentir OntoRev lorsqu'il fonctionne sur des ontologies contenant une quantité importante de non-déterminisme.

La structure de donnée la plus importante de OntoRev est TREENODE dont les objets peuvent être combinés dans un arbre. Un tel arbre n'est pas un arbre spécial tel qu'un arbre équilibré. Il peut avoir n'importe quel nombre de niveaux et chaque noeud peut avoir n'importe quel nombre d'enfants. Chaque noeud de l'arbre peut avoir au plus un parent et de 0 à plusieurs enfants. TREENODE fournit des opérations pour

Algorithme 1 : Algorithme de tableau vérifiant la cohérence d'une ontologie avec individus en $\mathcal{SHIQ}$

Entrée : $\mathcal{O}$: une ontologie avec individus en $\mathcal{SHIQ}$
Sortie : Est-ce que $\mathcal{O}$ est cohérente ?

```
1  Initialiser un ensemble de forêts ForestsSet = InitForests(O);
2  tant que ForestsSet ≠ ∅ faire
3      Retirer une forêt F de ForestsSet ;
4      si F est non-clash et complète alors
5          retourner OUI;
6      tant que il existe un nœud x de F qui n'est pas bloqué et il y a une règle r
         qui est applicable sur x faire
7          si r est une des ∃-, ∀-, ∀+- et ≥-règles alors
8              Appliquer r ;
9          si r est la ≤-règle (resp. la ≤r-règle) alors
10             pour chaque couple de voisins yi, yj de x qui satisfait des
                 conditions de la ≤-règle (resp. la ≤r-règle) faire
11                 Créer une copie F' de F ;
12                 Fusionner y'i avec y'j sur F' comme le comportement de la
                     ≤-règle (resp. la ≤r-règle) où y'i, y'j se trouvant sur F' sont
                     des copies de yi, yj respectivement ;
13                 Ajouter F' dans ForestsSet ;
14             Revenir à la ligne 2 ;
15         si r est la sat-règle alors
16             pour chaque ensemble S ⊆ sub(O) qui satisfait des conditions de
                 la sat-règle faire
17                 Créer une copie F' de F ;
18                 Ajouter S et S̄ dans L(x') sur F' où x' se trouvant sur F' est la
                     copie de x et S̄ = {¬̇C | C ∈ sub(O) \ S} ;
19                 Ajouter F' dans ForestsSet ;
20             Revenir à la ligne 2 ;
21 retourner NON;
```

examiner et modifier les parents et les enfants d'un noeud. Le sous-arbre enraciné en un noeud est l'ensemble de tous les noeuds qui peuvent être atteints en commençant par ce noeud et en suivant tous les liens possibles vers parents et enfants. Un noeud sans parent est la racine de son arbre ; un noeud sans enfant est une feuille. Un arbre peut être constitué de plusieurs sous-arbres ; et, chaque noeud agit comme la racine pour son propre sous-arbre.

A chaque objet de TREENODE est également assigné des attributs de validation, de saturation et d'application de règle de saturation , dite sat-règle. L'algorithme de

tableau applique des règles possibles sur chaque objet de TREENODE pour construire des arbres ou des forêts. De plus, en appliquant les techniques d'optimisation lors de la construction de $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$, l'étiquette de chaque nœud (objet de TREENODE) d'une forêt dans $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$ doit être validée, saturée et la sat-règle appliquée si nécessaire.

Les étapes effectuées dans ONTOREV sont illustrées dans la figure 5. Il y a quatre fonctions principales définies comme suit : CHO - chargeur d'ontologies ; CTM - constructeur de modèles de forêt ; CAR - calculateur de modèles résultants de révision ; et CTO - constructeur de l'ontologie de révision. Tout d'abord, la fonction CHO charge l'ontologie initiale et la nouvelle ontologie à réviser. Ensuite, les modèles de forêt et les modèles résultants de la révision (*i.e.*, $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$) sont construits par les fonctions CTM et CAR respectivement.

Enfin, CTO construit l'ontologie de révision à partir de l'ensemble des forêts $\mathsf{FM}(\mathcal{O}, \mathcal{O}')$. Nous allons présenter quelques expérimentations dans la section suivante.

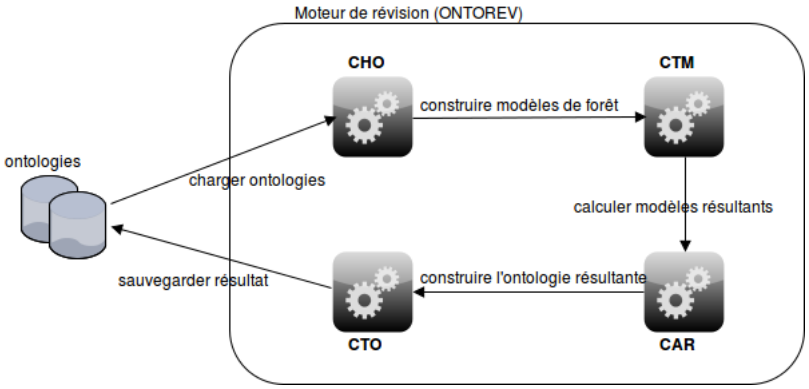

Figure 5. Etapes effectuées dans ONTOREV

6. Utilisation en ligne d'*OntoRev*

Un module (*cf.* le rectangle « Moteur de révision » dans la figure 6) chargé de la révision d'ontologie (ONTOREV) est intégrable dans toute plateforme (par exemple, dans la plateforme *LearningCafé* qui concernait l'apprentissage de gestes techniques par vidéo lors du projet FUI15). Ce module reçoit une requête de révision de l'interface de la plate-forme (*cf.* la flèche 1.2 dans la figure 6) et effectue la révision sur les ontologies concernées de la plate-forme. Une telle révision peut entraîner une modification des ontologies en assurant leur cohérence. Un message est renvoyé à la plate-forme pour informer du résultat de la requête de révision. Tous ces échanges s'effectuent par des services Web et sont illustrés dans la figure 6.

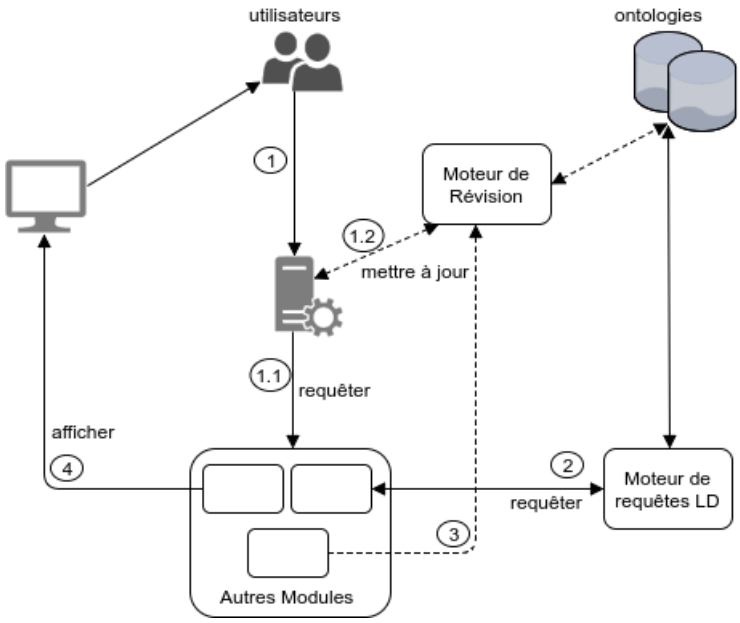

Figure 6. Echanges de données entre le module de révision et d'autres modules de la plateforme LearningCafé

Nous avons implémenté et mis en ligne ONTOREV [1] un module simple pour la mise à jour des connaissances modélisées dans les ontologies.

La page d'accueil (*cf.* figure 7) est découpée en trois zones :

– **Ontology Information** : Un utilisateur peut charger une ontologie en tapant un lien [2] 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.

– **Toolbox** : contient des boutons qui permettent de charger une ontologie (**Load**), de sauvegarder le résultat (**Save**), ou d'annuler le traitement (**Cancel**). De plus, l'utilisateur peut cliquer sur **More Infos** pour obtenir plus d'information sur l'ontologie en cours de traitement. Le bouton **Add New Infos** lui permet de mettre à jour l'ontologie. Enfin, il peut aussi faire des requêtes sur l'ontologie en utilisant le bouton **DL Query & Entailment**.

– **Console** : affiche une notification lors d'une action qui est réalisée sur le moteur.

1. http://linc.iut.univ-paris8.fr:8080/search-revision-engine/
2. sous la forme de http://exemple.com/url/vers/votreOntologie.owl

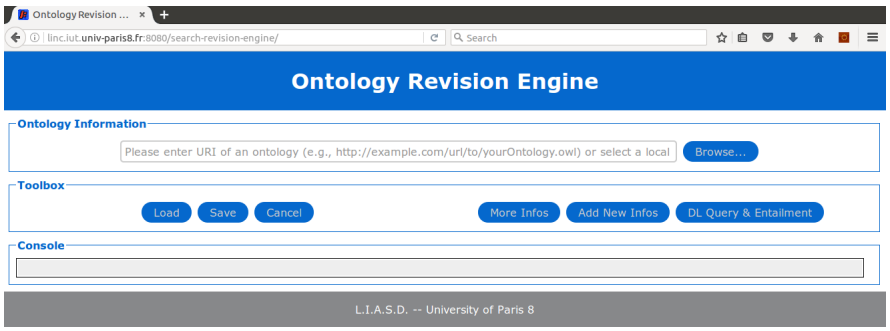

Figure 7. Page d'accueil

En outre, une fois qu'une ontologie est chargée, notre moteur de révision affichera de nouveaux boutons (*cf.* figure 8) qui permettent d'ajouter de nouvelles entités, de nouveaux axiomes ou de nouvelles assertions.

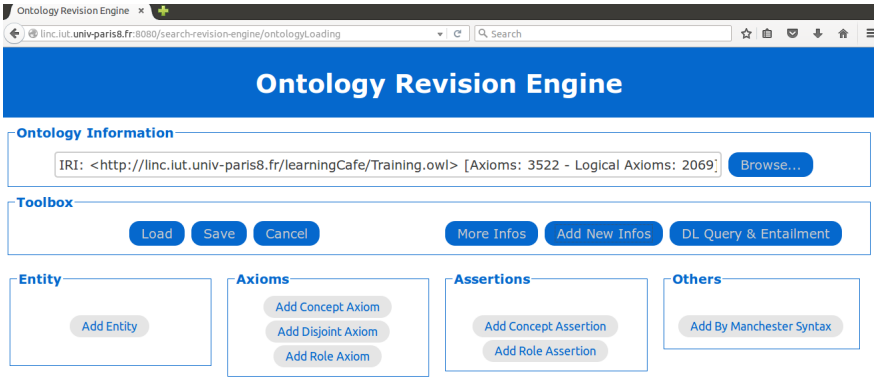

Figure 8. Outils pour la révision d'une ontologie

6.1. Scénario avec ontologie révisée cohérente

Ce premier scénario va consister à ajouter un nouveau concept dans l'ontologie *Training.owl* utilisée lors du projet *LearningCafé* dont quelques axiomes ont été vus dans l'exemple 1. Pour ce faire, cliquons sur le bouton **Add New Infos** et puis **Add Entity** (*cf.* figure 9) pour ajouter une nouvelle classe de formation en peinture, soit une nouvelle entité nommée **PainterTraining** ; choisissons un type **Class** et cliquons sur le bouton **Add** (*cf.* figure 10).

La nouvelle classe **PainterTraining** a été ajoutée dans l'ontologie qui est toujours cohérente (*cf.* figure 11).

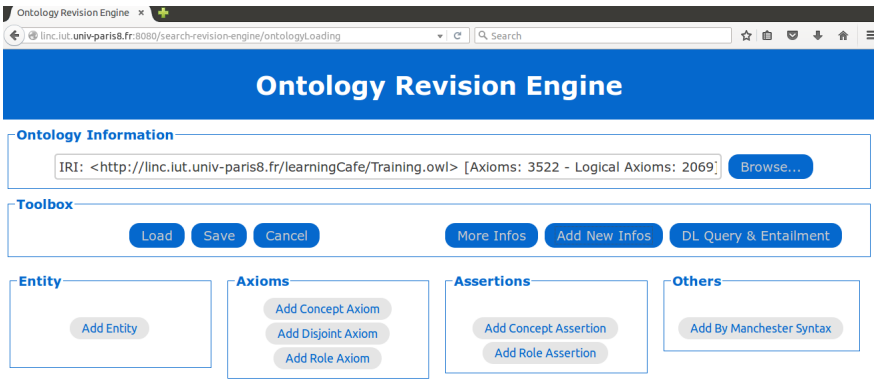

Figure 9. Boutons pour la mise à jour d'une ontologie

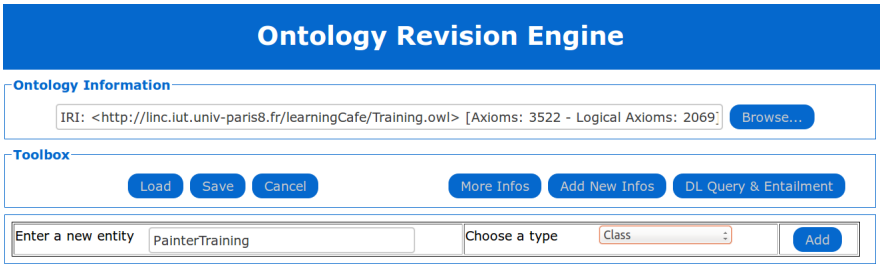

Figure 10. Ajout d'une nouvelle classe PainterTraining

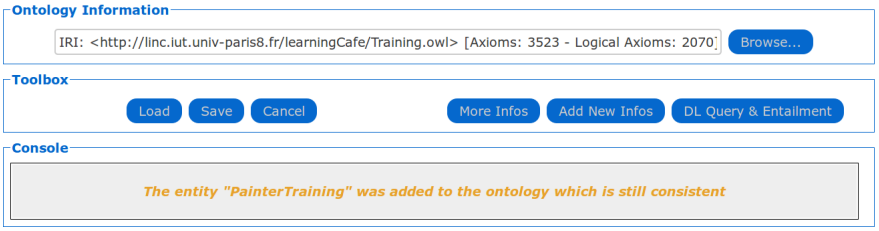

Figure 11. Résultat de l'ajout d'une nouvelle classe PainterTraining

6.2. Scénario avec ontologie révisée incohérente

De la même façon que dans la sous-section 6.1, ajoutons une nouvelle classe **BakerPastryTraining**, indiquant une classe de formations de boulangerie-pâtisserie (*cf.* figure 12).

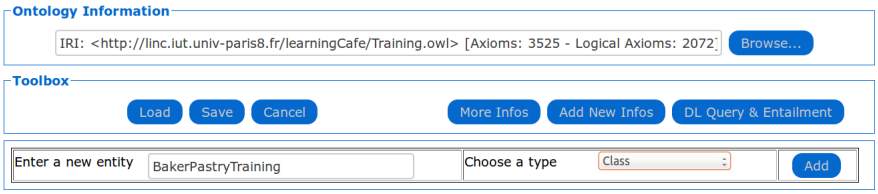

Figure 12. Ajout d'une nouvelle classe BakerPastryTraining

Cliquons sur le bouton **Add New Infos** et puis **Add Concept Axiom** (*cf.* figure 12) pour ajouter un axiome disant qu'une classe de formations de boulangerie-pâtisserie est une sous-classe d'une classe de formations de boulangerie. Pour ce faire, on choisit la classe **BakerPastryTraining** qui est une sous-classe de la classe **BakerTraining** comme dans la figure 13.

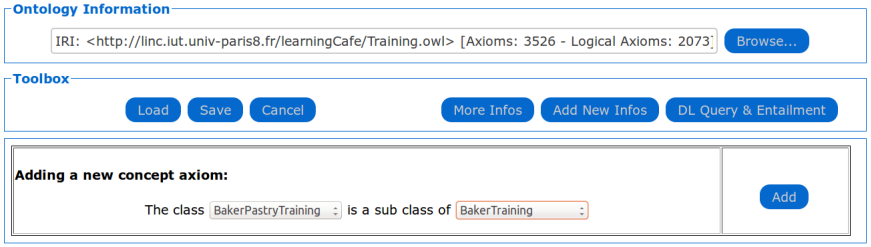

Figure 13. Ajout d'un nouvel axiome

Maintenant, ajoutons un nouvel individu **BakerPastryTraining_Course1_2015**, qui déclare une instance de la formation de boulangerie-pâtisserie, en faisant de la même manière que lors de l'ajout d'un individu (*cf.* figure 14).

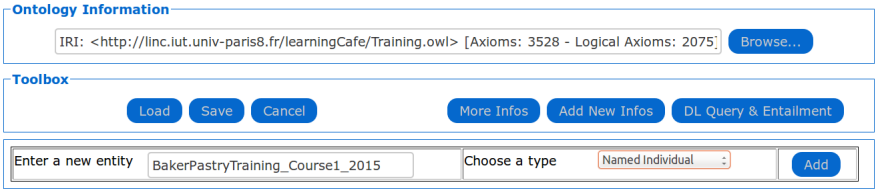

Figure 14. Ajout d'un nouvel individu

Maintenant, ajoutons le nouveau fait à savoir l'individu

BakerPastryTraining_Course1_2015 (une formation) est une instance de la classe **BakerPastryTraining** (une classe de formations de boulangerie-pâtisserie), en faisant de la même manière que lors de l'ajout d'une assertion (*cf.* figure 15).

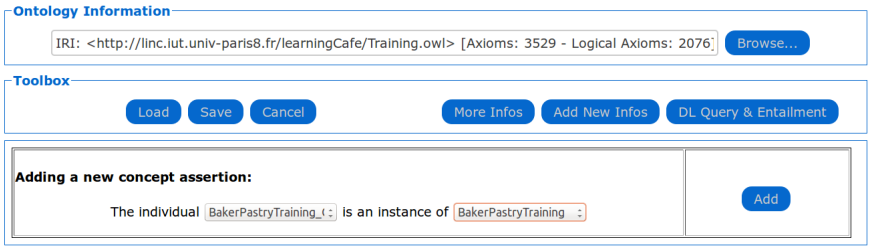

Figure 15. Ajout d'une nouvelle assertion (un fait)

La console affiche une notification disant que l'ontologie devient incohérente après l'ajout de cette assertion car la formation **BakerPastryTraining_Course1_2015** est une instance de la classe **BakerPastryTraining** de formations de boulangerie-pâtisserie. De plus, la classe **BakerPastryTraining** est une sous-classe de **BakerTraining** et de **PastryTraining**. En revanche, les classes **BakerTraining** et **PastryTraining** sont déclarées disjointes, c'est-à-dire qu'elles ne peuvent pas contenir une même instance (*cf.* figure 16).

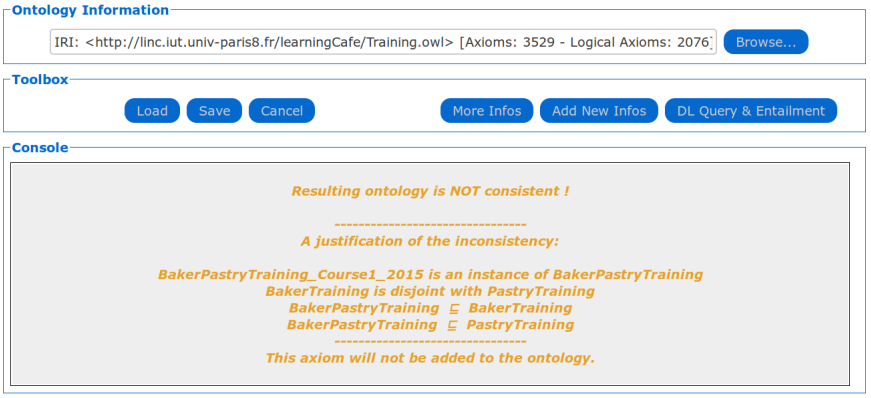

Figure 16. Résultat de l'ajout d'une assertion qui rend l'ontologie incohérente

7. Discussion

L'intelligence collective porte des dimensions psycho-sociologique en plus de la dimension collaborative (Kolonin *et al.*, 2016). Dans cet article, nous nous attachons surtout à la dimension collaborative par l'intermédiaire de l'outil en ligne OntoRev. Non seulement OntoRev détermine en temps réel les possibilités d'incohérence sur le savoir et le savoir-faire modélisés mais aussi est capable d'inférer des connaissances cachées. Par exemple, ces nouvelles connaissances révélées aux membres d'un groupe

de travail peuvent faciliter la prise de décision. En ce sens, OntoRev se comporte comme un « membre » du groupe capable de distinguer des informations erronées et des informations tacites. Une expérimentation lors du projet *Learning Café* lors des activités de groupe dans les phases d'apprentissage de gestes techniques pour des métiers manuels a montré qu'il était utile pour valider ou non certains choix des apprenants. L'intelligence collective se met alors au service de l'intelligence individuelle pour améliorer le savoir et/ou le savoir-faire de chacun.

Cependant, l'augmentation de la capacité à innover ou à prendre des décisions dans un laps de temps donné d'un groupe de travail utilisant notre outil ne peut être démontrée s'il est utilisé tout seul. En effet, la constitution du groupe est prépondérante, notamment par rapport à sa dimension sociale. Des travaux de recherche expérimentale indiquent seulement les caractéristiques de « constitution » d'un groupe qui, combiné avec des processus d'interaction peuvent être source d'intelligence collective (Woolley *et al.*, 2015). Ce cas a été aussi expérimenté dans le cadre du projet *Learning Café* en couplant OntoRev avec un système multi-agents représenté par la partie « autres modules » dans la figure 6) qui est en charge de la constitution des groupes d'apprenants. Ici, le pilotage des agents s'aidant d'OntoRev, des profils des différents membres du groupe et des solutions envisagées par d'autres groupes lors des activités collaboratives pour la formation à des métiers manuels a permis de réduire le temps de compréhension et donc d'acquisition de savoir et de savoir-faire.

D'autre part, si l'on se réfère au fait que la coopération prend tout son sens quand le problème à résoudre est complexe, qu'il existe une forte corrélation entre les capacités de perception sociale et la performance du groupe dans des environnements en ligne permettant aussi l'utilisation de repères issus du langage non verbal (Engel *et al.*, 2014), il devient évident que notre outil doit être intégré dans des plateformes plus complètes permettant des visio-conférences ou des audio-conférences entre autres.

8. Conclusion et perspectives

Il n'est plus à démontrer la nécessité d'utiliser l'intelligence collective lors du fonctionnement d'un groupe de travail que ce soit pour la formation, la prise de décision ou l'innovation. Pour ce faire, les outils numériques intègrent de plus en plus un niveau sémantique élevé, notamment par des représentations ontologiques.

Nous avons implémenté et testé un prototype OntoRev pour la révision d'ontologies en $\mathcal{SHIQ}$. Les résultats expérimentaux obtenus ont indiqué que notre algorithme / implémentation n'est pas encore assez performant sur des ontologies qui contiennent une quantité importante de non-déterminisme résultant de disjonctions, de restrictions de cardinalité. Pour résoudre ces inconvénients, nous chercherons à concevoir un algorithme de tableau ExpTime pour vérifier la cohérence des ontologies en $\mathcal{SHIQ}$ car un tel algorithme nous permettrait de réduire non seulement le non-déterminisme, mais aussi la profondeur des forêts de complétion. Bien entendu, en soi, la performance de notre algorithme est transparente aux utilisateurs finaux et ne peut gêner que dans le cas de la nécessité d'aider la prise de décision en temps réel.

Nous avons aussi implémenté et exécuté en ligne [3] un moteur (intégrant le prototype OntoRev) pour la mise à jour des connaissances modélisées dans les ontologies. Tel quel, OntoRev est essentiellement un soutien et un garde-fou notamment quand un projet à mener est très complexe et que les membres du groupe de travail ne peuvent avoir une vue d'ensemble de ce dernier. Nous avons donc fourni un ensemble de services Web qui permettent d'intégrer notre outil dans des plateformes comportant d'autres outils collaboratifs favorisant l'intelligence collective de groupes de travail. Il serait intéressant maintenant de fournir un minimum de trace sur les approximations sémantiques faites lors de la révision pour faciliter un certaine forme de négociation entre les membres du groupe de travail afin d'aboutir plus rapidement à un consensus de modélisation ou à une meilleure compréhension de la cartographie des connaissances.

Remerciements

Ce travail a été financé grâce au FUI15-LearningCafé

Bibliographie

Alchourrón C., Gärdenfors P., Makinson D. (1985). On the logic of theory change : Partial meet contraction and revision functions. *Journal of symbolic Logic*, vol. 50, p. 510-530.

Baader F., Horrocks I., Sattler U. (2003). Description logics as ontology languages for the semantic web. In *Festschrift in honor of jörg siekmann, lecture notes in artificial intelligence*, p. 228–248. Springer-Verlag.

Baader F., Nutt W. (2003). Basic description logics. In *The description logic handbook : Theory, implementation and applications*, p. 47–100. Cambridge University Press.

Baader F., Sattler U. (2001). An overview of tableau algorithms for description logics. *Studia Logica*, vol. 69, n° 1, p. 5–40.

Dalal M. (1988). Investigations into a theory of knowledge base revision. In *Proceedings of aaai*, p. 475–479.

De Giacomo G., Lenzerini M., Poggi A., Rosati R. (2007). On the approximation of instance level update and erasure in description logics. In *Proceedings of AAAI*, p. 403–408.

Eiter T., Gottlob G. (1992). On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, vol. 57, p. 227–270.

Engel D., Woolley A. W., Jing L. X., Chabris C. F., Malone T. W. (2014). Reading the mind in the eyes or reading between the lines? theory of mind predicts collective intelligence equally well online and face-to-face. *Plos One*, vol. 9, n° 12, p. e115212.

Fournier-Viger P. (2005). Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents. In *Mémoire de maîtrise (m.sc.), université de sherbrooke, sherbrooke, canada.*

3. http://linc.iut.univ-paris8.fr:8080/search-revision-engine/

Glimm B., Horrocks I., Motik B. (2010). Optimized description logic reasoning via core blocking. In *Proceedings of the 5th international conference on automated reasoning*, p. 457–471.

Grau B. C., Jimenez-Ruiz E., Kharlamov E., Zheleznyakov D. (2012). Ontology Evolution Under Semantic Constraints. In *Proc. of international conference on principles of knowledge representation and reasoning (kr)*, p. 137-147.

Hansen M. J., Vaagen H. (2016). Collective intelligence in project groups: Reflections from the field. *Procedia Computer Sciences, Elsevier*, vol. 100, p. 840–847.

Horrocks I., Sattler U., Tobies S. (1999). Practical reasoning for expressive description logics. In *Logic programming and automated reasoning, 6th international conference, lpar'99, tbilisi, georgia, september 6-10, 1999, proceedings*, p. 161–180.

Katsuno H., Mendelzon A. O. (1991). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, vol. 53, n° 3, p. 263–294.

Kharlamov E., Zheleznyakov D., Calvanese D. (2013, septembre). Capturing model-based ontology evolution at the instance level: The case of dl-lite. *J. Comput. Syst. Sci.*, vol. 79, n° 6, p. 835–872.

Kolonin A., Vityaev E., Orlov Y. (2016). Cognitive architecture of collective intelligence based on social evidence. *Procedia Computer Sciences, Elsevier*, vol. 88, p. 475–481.

Krötzsch M., Vrandecic D. (2011). Semantic mediawiki. In *Foundations for the web of information and services*, p. 311–326. Springer.

Lévy P. (1994). *L'intelligence collective. pour une anthropologie du cyberespace*. La Découverte, Paris.

Maleszka M., Nguyen N. T. (2015). Integration computing and collective intelligence. *Elsevier Expert Systems with Applications*, vol. 42(1), p. 332–340.

Minsky M. (1981). A framework for representing knowledge. In J. Haugeland (Ed.), *Mind design: Philosophy, psychology, artificial intelligence*, p. 95–128. Cambridge, MA, MIT Press.

Qi G., Du J. (2009). Model-based revision operators for terminologies in description logics. In *Proceedings of the 21st international jont conference on artifical intelligence*, p. 891–897. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc.

Qi G., Wang Z., Wang K., Zhuang X. F. Z. (2015). Approximating model-based abox revision in dl-lite: Theory and practice. In *Proceedings of the twenty-ninth aaai conference on artificial intelligence*, p. 254–260. AAAI Press.

Satoh K. (1988). Nonmonotonic reasoning by minimal belief revision. In *Proceedings of the international conference on fifth generation computer systems*, p. 455–462.

Sattler U. (1996). A concept language extended with different kinds of transitive roles. In G. Görz, S. Hölldobler (Eds.), *20. deutsche jahrestagung für künstliche intelligenz*, p. 333–345. Springer Verlag.

Shearer R., Motik B., Horrocks I. (2008, October 26–27). Hermit: A highly-efficient owl reasoner. In A. Ruttenberg, U. Sattler, C. Dolbear (Eds.), *Proc. of the 5th int. workshop on owl: Experiences and directions (owled 2008 eu)*. Karlsruhe, Germany.

Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, n° 2, p. 51 - 53.

Tobies S. (2001). Complexity results and practical algorithms for logics in knowledge representation. *PhD thesis, RWTH Aachen, Germany*.

Trappey C. V., Smith P., Trappey S., Chen L. W. L., Tung J. T. C. (2014). Using the collective intelligence of sports fans to improve professional football league customer service. In *CSCWD*, p. 313–318. IEEE.

Tsarkov D., Horrocks I. (2006). Fact++ description logic reasoner: System description. In *Proceedings of the third international joint conference on automated reasoning*, p. 292–297.

Wang Z., Wang K., Topor R. (2010). A new approach to knowledge base revision in dl-lite. In *Proceedings of the twenty-fourth conference on artificial intelligence, aaai 2010, atlanta, georgia, usa, july 11-15*, p. 369–374.

Woolley A., Agarwal I., Malone T. (2015). Collective intelligence and group performance. , vol. 24, n° 6, p. 420-424.