Intégration de raisonnements automatiques dans le système d'annotation MemoNote

Hakim Mokeddem, Cyrille Desmoulins

Intégration de raisonnements automatiques dans le système d'annotation MemoNote

Hakim Mokeddem[1], Cyrille Desmoulins[2]

[1] Ecole Supérieure d'Informatique ESI, Alger
h_mokeddem@esi.dz
[2] Laboratoire d'Informatique de Grenoble, Université Joseph Fourier,
cyrille.desmoulins@imag.fr

Résumé : Les raisonnements automatiques dans les EIAH basés sur les ontologies reposent le plus souvent sur des algorithmes ad hoc du fait des langages de représentations utilisés. Cet article développe autour de l'exemple du logiciel d'annotation MemoNote, les avantages de l'utilisation des technologies standards du web sémantique pour permettre des raisonnements sur les ontologies. A partir des limites de la version MemoNote actuelle basée sur le langage des Frames, l'utilisation d'une représentation OWL et de raisonneurs standards est développée sur trois exemples de requêtes. L'intégration de ces technologies permet d'étendre les fonctionnalités de MemoNote tout en améliorant l'expressivité, l'interopérabilité, la complétude et la correction des raisonnements. La complexité reste à étudier sur des masses importantes de données et pourrait nécessiter le recours à des SGBD sémantiques, garantissant en même temps les propriétés ACID.

Mots-clés : Annotation sémantique, ontologie, raisonnements automatiques, raisonneur, web sémantique, OWL.

1 Introduction

L'annotation est l'une des activités les plus pratiquées parmi les enseignants sur les documents papier. Elle lui sert de mémoire externe de ses points de vue et opinion sur les éléments du document et soulage sa propre mémoire (Sperber 2001).

Avec la grande disponibilité des supports pédagogiques électroniques sur le web, l'annotation électronique devient nécessaire comme sur le papier. Pour pouvoir être traitée par des programmes informatiques, elle doit reposer sur une structure formelle tout en conservant la facilité d'annotation du papier.

Différents outils d'annotations sont disponibles, dans le commerce ou issus de travaux de recherche. Certains visent uniquement à aider l'annotateur à utiliser une forme graphique pour ajouter des commentaires textuels sur le document, comme Microsoft Word et iMarkup (iMarkup-Solutions-Inc 2004). D'autres permettent l'interprétation de la sémantique des annotations par moyens automatiques, par exemple Annotea (KAHAN et al. 2001). Enfin des outils comme Melita (Ciravegna et al. 2002) automatisent le processus d'annotation par apprentissage automatique d'après les actions de l'utilisateur.

Tous ces outils sont génériques et ne permettent pas de supporter les activités spécifiques d'annotation de l'enseignant. En particulier, ils ne reposent pas sur une sémantique des points de vues pédagogiques de l'enseignant, rendant nécessaire la conception d'un outil d'annotation dédié à l'enseignant. C'est la raison du développement de MemoNote (Azouaou & Desmoulins, 2006). Par l'usage de patrons d'annotation, il facilite la création efficace d'annotations dans un contexte pédagogique donné. En même temps, il mémorise une sémantique accessible aux programmes informatiques, par le biais d'une ontologie des annotations intégrant les points de vue de l'enseignant. Il permet ainsi la recherche avec cette sémantique pédagogique et la notification automatique à l'enseignant d'annotations de rappel en fonction du contexte courant.

Cependant, la version actuelle de MemoNote est contrainte par des limites liées aux technologies non standards sur lesquelles il repose. En effet, MemoNote utilise la version de Protégé 3.x fondée sur des Frames (Kifer & Lausen, 1995) pour représenter les ontologies, ce qui limite l'interopérabilité avec d'autres systèmes. Ce langage ne garantit pas la correction des réponses aux requêtes faites sur l'ontologie, ni leur complexité, ce qui entraine des performances variables. La complétude des réponses n'est pas non plus assurée avec ce langage. A tous ces problèmes s'ajoute le problème des fonctionnalités limitées de la version actuelle de MemoNote.

L'objectif de cet article est d'étudier l'intégration des technologies standards du web sémantique (Berners-Lee 1998) comme OWL (McGuinness & van Harmelen, 2004) pour modéliser l'ontologie annotation afin de l'utiliser avec des systèmes de raisonnements standards et ainsi d'obtenir l'interopérabilité de MemoNote, la complétude et la correction des réponses ainsi que des bonnes performances du système.

Pour atteindre ces objectifs, nous étudions en détail les limites de MemoNote et des autres systèmes pédagogiques utilisant de raisonnements automatiques sur des ontologies. Nous décrivons ensuite les besoins fonctionnels de l'enseignant en termes de cas d'utilisations qui montrent la nécessité de raisonnements automatiques. Pour implanter les requêtes correspondant à ces besoins, nous développons l'utilisation de langages standards du Web sémantique. Enfin nous évaluons cette

utilisation de standards de représentation et de raisonnements automatiques.

2 Limite des systèmes pédagogiques basés sur des raisonnements sur les ontologies.

MemoNote mémorise les annotations à partir d'une ontologie des annotations. Cette ontologie permet de représenter les éléments l'annotation de l'enseignant suivants :

- Le contexte de l'annotation est représenté par les données de la séance d'annotation, qui est décrite par l'enseignant annotateur, le lieu d'annotation, les dates de début/fin, le domaine et le niveau d'apprentissage, les types d'activités d'enseignements et d'apprentissages.
- L'épisode d'annotation comprend l'ancre d'annotation dans le document et sa date.
- La forme graphique comprend la forme graphique et la couleur de l'annotation (voir Fig 1).
- L'objectif de l'annotation est décrit par son type, sa force et des éléments liés à son type.

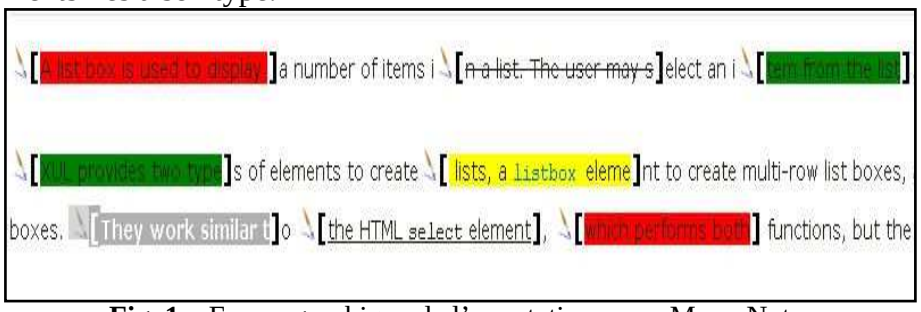

Fig. 1 – Forme graphique de l'annotation avec MemoNote

Pour automatiser l'annotation, MemoNote utilise des patrons d'annotation qui permettent de déduire la sémantique de l'annotation à partir de son contexte et de sa forme graphique (voir Fig 2). Les patrons sont eux aussi représentés par une ontologie qui réfère aux éléments de l'ontologie d'annotation.

Fig. 2 – Barre d'outils MemoNote

Le langage de Frame de la version actuelle de MemoNote permet de représenter les éléments de ces ontologies sous forme de classes et propriétés (slots) qui décrivent les classes et les relations entre ces classes. Cependant aucun raisonneur standard n'est fourni avec ce langage, ce qui nécessite le développement de programmes de

raisonnement ad hoc. Aucun résultat théorique n'existe sur ce type de langage, ce qui ne permet pas de garantir la correction, la complétude voire même les performances de ces programmes complexes à développer.

En général, les raisonnements automatiques ont été utilisés dans plusieurs domaines. Par exemple, dans le domaine médical, les travaux de (Levy et al.2009*)* montrent l'utilisation des raisonnements automatiques pour la recherche des lésions cancéreuses dans le corps humain. Les chercheurs proposent au début de transformer les annotations des images radiologiques décrites avec AIM (Annotation and Image Markup) en OWL et présentent ensuite un cas l'utilisation du raisonnement automatique à travers un cas d'utilisation.

Dans le domaine des EIAH, d'autres recherches utilisent les raisonnements automatiques sur les ontologies. (Henze et al. 2004) utilisent des déductions automatiques pour un système hypermédia adaptatif d'apprentissage en ligne. Les déductions sont utilisées pour personnaliser les ressources présentées à l'apprenant. Il utilise des ontologies RDF (Lassila & Swick, 1999) pour représenter le domaine, le modèle de l'apprenant à partir de l'observation de son activité sur les ressources. Pour raisonner, le système utilise le langage de règle TRIPLE RULE LANGUAGE (Decker et al. 2005). L'inconvénient de ce langage est l'écriture manuelle de toutes les règles pour assurer l'inférence. L'autre inconvénient est la difficulté d'exprimer des règles complexes avec ce langage. Par exemple, pour adapter le contenu à l'apprenant, le système doit afficher tous les bons exemples des concepts d'un cours, que la règle recherchera pour extraire les ressources en relation avec ces exemples positifs.

Les recherches de (Huang et al.2006) ont abouti à l'implémentation d'un système de recherche sémantique des contenus pédagogiques d'un système d'apprentissage en ligne à partir du contexte. Le contexte est modélisé avec le langage RDF sous forme de triplet (sujet, prédicat, objet). L'utilisateur effectue la recherche à partir du contexte en spécifiant des paires <prédicat, objet>. Le système extrait ensuite le contenu approprié à ce contexte. Par exemple pour rechercher un contenu à partir du contexte, l'utilisateur choisi comme objet différents contextes (université, forum, …), et le système lui affiche le contenu qui correspond exactement à ce contexte. Aucun raisonnement automatique n'est effectué, et les ressources retrouvées sont assez pauvres.

Certains outils sont basés sur des langages standards ISO comme TM4L (Dicheva & Dichev, 2006) basé sur les Topic Maps. Ils permettent de représenter des ontologies du domaine par des sujets (Topics) et des objets (instances de Topics), en utilisant le standard XTM (XML Topic Maps). Ce langage est cependant très peu utilisé à cause de la non disponibilité de raisonneurs standards comme c'est le cas pour les standards du W3C (OWL).

A partir de ces études, nous constatons que l'utilisation des raisonnements avec les technologies du web sémantiques dans les outils d'apprentissage est limitée. Certains sont basés sur des règles ad hoc alors que d'autres offrent des requêtes simples qui ne prennent pas en compte toute la sémantique des ontologies. Et pourtant, comme nous montrons dans la section suivante, les raisonnements automatiques sur l'annotation et son contexte sont nécessaires pour fournir des services profitant pleinement de la sémantique formelle des annotations. Notre objectif est donc d'étudier les capacités de raisonnements automatiques avec les technologies standards pour les besoins de l'outil d'annotation MemoNote et les problèmes que pose l'implantation de ces types d'inférences.

3 Les besoins en raisonnement

Pour définir les besoins en raisonnement, nous présentons maintenant des cas d'utilisations améliorant les fonctionnalités de MemoNote.

Recherche d'annotations à partir d'un contexte.
Cette recherche permet à l'enseignant de retrouver les annotations produites dans un contexte sémantiquement équivalent à un contexte donné. Par exemple l'enseignant recherche les annotations produites par Paul tel que le lieu est le campus de Grenoble. Cela comprend les annotations produites dans tous les lieux inclus dans le campus de Grenoble.

Recherche d'annotations à partir d'un objectif.
Pour chaque annotation, l'enseignant renseigne le but pour lequel l'annotation est créée. Il peut ensuite retrouver les annotations qui ont un objectif similaire à objectif donné. Par exemple, il peut rechercher les annotations par Paul telles que l'objectif de l'annotation est la vérification d'un calcul chimique. Un raisonneur lui permet de retrouver les annotations avec un objectif similaire, ou plus général.

Activation de patrons d'annotation
L'utilisation des patrons d'annotation permet à l'enseignant d'automatiser le processus d'annotation. Ainsi l'enseignant ne renseigne pas à chaque fois ses objectifs d'annotation, le patron le prenant en charge à partir du contexte et de la forme choisie. Ainsi MemoNote doit être capable de d'activer un patron en fonction du contexte et de la forme graphique, pas seulement en cas de correspondance exacte, mais aussi quand la séance est un cas particulier du contexte du patron.

Dans la partie suivante nous décrivons les raisonnements automatiques sur l'ontologie d'annotation nécessaires à ces cas d'utilisations et leur implantation avec des technologies du web sémantique.

4 Raisonnements automatiques

Nous commençons par présenter le modèle des annotations sur lequel reposent les raisonnements automatiques.

Les annotations sont représentées par la classe *Annotation*. Cette classe possède trois sous-classes pour chacun des types d'objectifs (domaine, document et pédagogique), elles-mêmes décomposées en sous-classes précisant les objectifs subjectifs de l'enseignant. C'est cette classification qui est spécifique à l'enseignant. Par exemple la classe *DomainAnnotation*, qui rassemble les annotations ayant un objectif au niveau du domaine d'enseignement, contient les sous-classes *Vérification d'un calcul chimique*, *Vérification d'un résultat de TP* et *Mémorisations d'une erreur possible*. Une annotation est reliée à la classe *Session* qui représente le contexte de l'annotation via la propriété *made_during* et chaque session est relié à la classe *Location* par la propriété *made_at_location.* Chaque lieu peut être inclue dans un autre par la propriété *is_part_of* (voir Fig 3).

Une annotation est aussi reliée à la classe *Visual_Shape* (formes graphiques) par la propriété *has_visual_shape,* elle est aussi reliée à la classe *Color* par la propriété hasColor. Chaque annotation possède une ancre physique représentée par la propriété Anchor et un créateur par la propriété *Author*. Une session a une date début (*beginDate)* et une date fin (*endDate)* (voir Fig 4).

Différentes classes représentent les éléments du contexte : le lieu de l'annotation est modélisé par la classe *Location* reliée à la classe *Session* par la propriété *made_at_location.* Chaque lieu peut être inclus dans un autre par la propriété *is_part_of. TeachingLevel* (par exemple License1, Master 2), *TeachingDomain* (Chimie, Mathématiques) et *TeachingActivityType* (Préparation TP, Préparation cours) permettent aussi de décrire une séance et sont reliées à la classe *Session.* Par exemple, on peut décrire une session d'annotation d'un enseignant dans le domaine de la chimie niveau Licence 1 et qui concerne l'activité de préparation d'un cours.

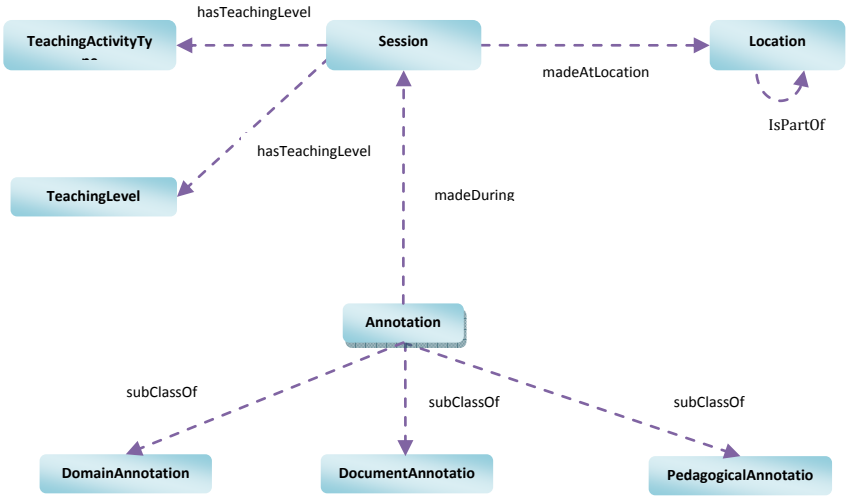

Fig. 3 – Classes principales de l'ontologie Annotation

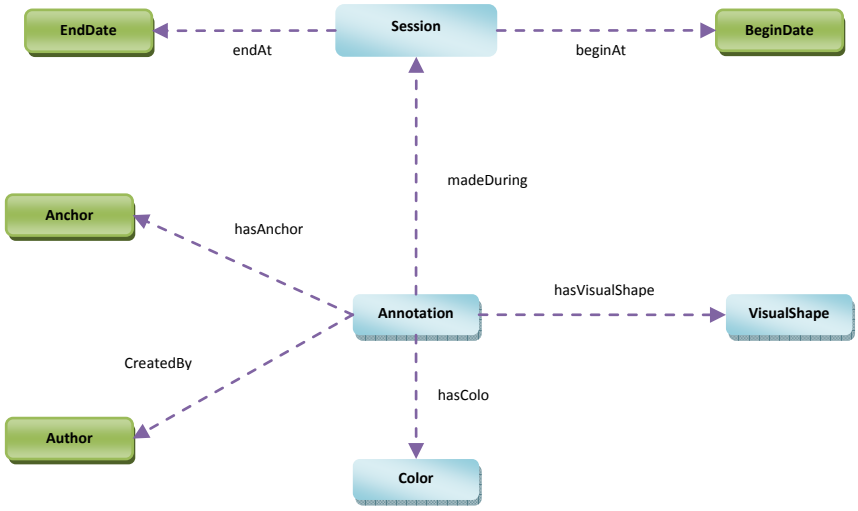

Fig. 4 – Classes secondaires de l'ontologie Annotation

Un patron d'annotation est représenté par la classe *AnnotationPattern.* Son contexte est représenté par une sous-classe particulière de *Session* et une instance de *Visual_Shape.* Cela permet d'accepter différentes instances de séances correspondant au contexte du patron (voir Fig5). Par exemple, on peut décrire le patron *ChemPattern* avec une classe contexte *ChemLabSession* qui décrit une annotation dans un laboratoire de chimie à l'université.

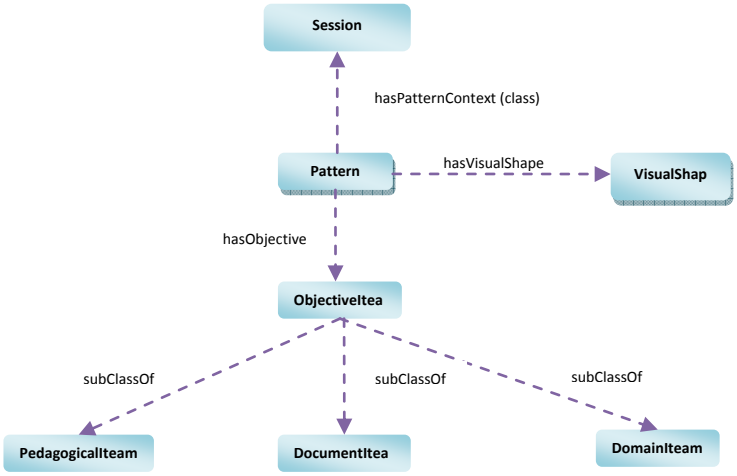

Fig. 5 – L'ontologie Patron d'annotation

Si l'on utilise les langages de requêtes RDF du web sémantique sur les instances de l'ontologie, on ne peut pas extraire toutes les instances d'annotations pour les cas d'utilisation décrits. Par exemple, pour répondre à la première requête, une requête SPARQL (Prud'hommeaux & Seaborne, 2008) donne comme résultat les annotations qui ont une relation *made_at_location* exactement égale à « Grenoble Campus » et non tous les lieux qui font partie du campus de Grenoble (par la relation *is_part_of*), les langages de requêtes utilise ce qui a été explicitement définis comme relations et ne raisonne pas pour extraire le modèle implicite. L'utilisation d'un raisonneur pour faire l'inférence devient alors une nécessité. Le raisonneur permet d'extraire de nouvelles relations entre les classes, et de déduire de nouveaux types d'instances à partir de la globalité de la définition de l'ontologie.

Nous détaillons maintenant les trois cas d'utilisations décrit précédemment en utilisant des raisonnements automatiques sur l'ontologie d'annotation en OWL.

Recherche d'annotations à partir d'un contexte.

L'enseignant recherche les annotations produites par Paul qui ont eu lieu dans le campus de Grenoble. Cela comprend les annotations produites dans tous les lieux inclus dans le campus de Grenoble.

Le langage Frame nécessite le développement d'un programme ad hoc calculant cette requête, selon l'algorithme suivant :

- une première étape extrait tous les lieux inclus dans le campus de Grenoble. On utilise une procédure récursive de parcours du graphe des inclusions par *is_part_of* à partir de « Grenoble Campus » pour obtenir la liste LI des lieux inclus.

Cela revient à reproduire par programme la transitivité de la propriété *is_part_of*.

- une deuxième étape construit la liste LS des sessions qui se sont déroulées dans les lieux de la liste LI. Ce sont toutes les sessions dont la valeur de *made_at_location* est dans LI.
- enfin la dernière étape retrouve les annotations produites durant ces sessions ; Ce sont toutes les séances dont la valeur de la relation *made_during* est dans LS.

Avec l'utilisation du standard OWL 1, les raisonneurs automatiques standards peuvent être utilisés pour remplacer ce programme ad hoc. Ils fournissent le modèle inféré à partir du modèle initial utilisant des axiomes et règles définit sur l'ontologie. Le model inféré contient ainsi toutes les relations initialement décrites plus les nouvelles.

Dans le cas d'OWL 1, il suffit

- de définir par un axiome que *is_part_of* est transitive.
- d'ajouter à l'ontologie une règle SWRL (Horrocks et al. 2004) définissant l'extension de la propriété *made_at_location*. Cette extension spécifie que *made_at_location* est étendue à toutes les sous-parties d'un lieu donné. Dans notre exemple, cela signifie que si une *Session* S a eu lieu dans une *Location* L et que L est une sous partie de « Grenoble campus », alors S a bien eu lieu dans « Grenoble campus ».

Pellet (Sirin et al. 2003) est alors capable de retrouver dans le modèle inféré de l'ontologie ainsi augmentée (T-Box et A-Box) toutes les instances d'annotations attendues, soit directement, soit par la règle SWRL et la transitivité de *is_part_of*.

Dans ce cas, bien que les raisonnements sur OWL 1 DL soient décidables, cette décidabilité n'est pas assurée à cause de l'introduction de règles SWRL.

OWL2 permet précisément d'éviter le recours aux règles SWRL et de conserver la décidabilité des calculs des raisonneurs. OWL 2 permet d'exprimer l'extension de propriété avec un *Property Chain Axiom* (Golbreich & Wallace, 2009). Une fois définie cette extension et la transitivité de *is_part_of*, tout raisonneur prenant en compte les *Property Chain Axioms* peut fournir directement les annotations sur l'ontologie (T-Box et A-Box). On profite alors de l'optimisation des calculs qu'apportent les raisonneurs, difficile à mettre en œuvre dans un programme ad hoc.

Recherche d'annotations à partir d'un objectif.

L'enseignant recherche les annotations de Paul telles que l'intention de l'annotation est la vérification d'un calcul chimique. Pour extraire toutes les annotations qui ont comme objectif la vérification d'un calcul chimique, deux cas sont possibles :

1. Soit l'annotation est l'instance directe de la classe « Vérification d'un calcul chimique » et aucun raisonnement n'est nécessaire.
2. Soit l'annotation est indirectement l'instance de la classe « Vérification d'un calcul chimique » et un

raisonnement est nécessaire sur les sous-classes et instance de la classe Annotation.

Avec le langage de Frame, un programme récursif ad hoc est nécessaire. Cependant, il ne permet pas, comme en OWL, de définir une classe par extension, et limite fortement l'expressivité des déclarations.

Avec OWL 1 (ou OWL 2), l'utilisation du raisonneur Pellet est directe, il suffit de classifier toutes les instances appartenant à la classe « Vérification d'un calcul chimique ».

Dans ce cas, les calculs sont décidables et optimisés par Pellet, contrairement à un programme ad hoc nécessaire avec le langage de Frame.

Activation automatique des patrons d'annotations

Pour activer automatiquement un patron, il faut être capable de vérifier si le contexte courant et la forme graphique choisie correspondent au patron, c'est à dire que la séance est un cas particulier du contexte du patron.

Le calcul de cette requête est similaire au précédent, avec deux cas :

1. Le contexte de la session est une instance directe de la classe définissant le contexte du patron, dans ce cas aucun raisonnement n'est nécessaire.
2. Le contexte de la session courante une instance indirecte de la classe définissant le contexte du patron, un raisonnement est nécessaire.

Avec le langage de Frame, un programme ad hoc est nécessaire. Il est plus complexe que le précédent, car il nécessite de considérer chaque élément du contexte courant (lieu, niveau d'enseignement, domaine d'enseignement et type d'activité d'enseignement, type d'activité d'apprentissage) pour tester récursivement s'ils sont sous-classe des éléments des définissant le contexte du patron. De plus, l'expression du contexte d'un patron est elle aussi plus complexe, car impossible à définir par une unique classe.

Avec OWL 1, l'utilisation du raisonneur Pellet est directe, il suffit de demander au raisonneur de retrouver le patron correspondant au contexte courant et à la forme graphique choisie. On note que l'ontologie des patrons n'est pas en OWL-DL mais en en OWL Full, car chaque instance de patron ne pointe pas vers une instance de session mais vers une classe qui décrit un contexte. Ce n'est pas un problème car aucun raisonnement n'est fait sur l'ontologie des patrons mais uniquement sur l'ontologie des annotations, qui est en OWL-DL. Pour effectuer le raisonnement, il suffit juste de vérifier que la session en cours est une instance de la classe *Session* du patron définit.

5 Implémentation avec Pellet et Jena et Protégé

Dans cette section, nous présentons un test d'implémentation du cas d'utilisation « recherche d'annotations à partir d'un contexte » avec l'API du web sémantique Jena (McBride 2007), le raisonneur Pellet et l'éditeur Protégé (Noy et al. 2000).Nous utilisons dans l'implémentation le langage OWL 1.

Nous utilisons l'éditeur Protégé pour définir la règle SWRL. Protégé possède une interface utile pour définir une règle SWRL, il l'inclut ensuite dans le fichier de l'ontologie OWL.

Jena est une API Java qui contient des classes et des interfaces pour l'interaction avec les modèles RDF et OWL et les raisonneurs comme Pellet. Jena permet la création, la mise à jour et la suppression des triplets RDF ainsi que l'interrogation basé sur SPARQL.

Jena interagit aussi avec le raisonneurs Pellet par deux manières différentes : soit en utilisant l'interface DIG (Turhan et al. 2008) ou l'interface fournie par Pellet. L'interface DIG est limitée car elle utilise un protocole standard pour communiquer avec n'importe quel raisonneur OWL-DL; elle fournit de ce fait moins de possibilités d'inférences contrairement à l'interface de Pellet. L'interface de Pellet n'a pas besoin aussi d'utiliser HTTP pour communiquer avec le raisonneur, ce qui n'est pas le cas de l'interface DIG. L'utilisation de la communication avec le protocole HTTP peut augmenter le temps de réponse des requêtes. Pour ces raisons, nous utilisons l'interface directe de Pellet pour communiquer entre Jena and Pellet.

Nous décrivons maintenant l'implémentation de deux cas d'utilisations présentés dans la section précédente.

- ***Recherche d'annotations à partir d'un contexte.***

Le code du programme qui répond à cette requête est simple. Il utilise une seule méthode pour extraire les annotations de Paul qui on eu lieu dans le campus de Grenoble et affiche les textes annotés. L'objet ontModel de type OntModel est responsable du chargement de l'ontologie en mémoire. On active l'option du raisonneur Pellet pendant la création de ontModel. Il permet d'inférer le modèle après la lecture par la méthode read(). Une requête SPARQL est lancée pour extraire les annotations de Paul dans le campus de Grenoble du modèle inféré (ontModel). (Voir Fig 6)

```
// Create the ontModel with Pellet reasoner option
OntModel ontModel=ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
ontModel.read(ontfile);
// Create a SPAQRL QUERY and execute it
String queryContent =
    "PREFIX base:<"+NS+">" +
    " select ?text WHERE { " +
    "?x base:annotatedText ?text."+
    "?x base:madeDuring ?y."+
    "?y base:annotator base:Paul."+
    "?y base:hasLocation base:Grenoble_campus }";

Query query = QueryFactory.create(queryContent);
QueryExecution qexec = QueryExecutionFactory.create(query,ontModel);
ResultSet rs = qexec.execSelect();
String text=null;
while(rs.hasNext()) {
    QuerySolution querySol = rs.next();
    RDFNode name = querySol.get("?text");
    text = name.toString();
    // Print the results
    System.out.println(text);
}
```

Fig. 6 –Programme pour raisonner sur le contexte

- ***Activation automatique des patrons d'annotations***

Dans ce cas, Pellet n'est pas utilisé pour inférer à partir de la classe Pattern mais à partir des classes du contexte, parce que l'ontologie des patrons est en OWL Full.

En premier lieu, le système crée un objet pattern de type OntClass, qui représente la classe Pattern. Il crée aussi deux objets : session de type Resource et hasContext de type Property. Ensuite, la requête SPARQL est lancée pour extraire l'objectif du patron s'il correspond au contexte courant. Cette requête SAPRQL vérifie si la session courante (par exemple session_96) est une instance de la classe pattern, et la forme graphique est égale à la forme graphique du patron (voir Fig 7).

```
// Create the ontModel with Pellet reasoner option
OntModel ontModel=ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
ontModel.read(ontfile);
// Get context pattern class
OntClass pattern = ontModel.getOntClass(NS+"Pattern");
Resource session = null;
Property hasContext = ontModel.getProperty(NS+"hasContext");
Iterator<?> i = pattern.listInstances();
while( i.hasNext() ) {
    Individual ind = (Individual) i.next();
    session = (Resource) ind.getPropertyValue(hasContext);
}
// Create a SPAQRL QUERY and execute it
String queryContent =
    " PREFIX base:<"+NS+">" +
    " PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
    " select ?objective WHERE { " +
    " base:session_96 rdf:type base:" + session.toString().substring(NS.length())+"."+
    " ?pattern base:hasVisualShape ?shape."+
    " Filter (?shape=base:underlining_r)." +
    " ?pattern base:hasObjective ?objective" +
    "}";

Query query = QueryFactory.create(queryContent);
QueryExecution qexec = QueryExecutionFactory.create(query,ontModel);
ResultSet rs = qexec.execSelect();
String obj=null;
while(rs.hasNext()) {
    QuerySolution querySol = rs.next();
    RDFNode name = querySol.get("?objective");
    obj = name.toString();
    // Print the results
    System.out.println(obj);
}
```

Fig. 7 –Programme du raisonnement sur l'activation des patrons.

6 Evaluation

Dans cette partie nous évaluons les solutions proposées dans cet article par rapport aux points soulevés dans l'introduction à savoir l'interopérabilité, la correction, la complétude, la complexité et les fonctionnalités limitées.

Contrairement à d'autres outils comme TM4L, notre système assure l'interopérabilité avec d'autres systèmes utilisant les technologies du W3C. En effet, les standards comme OWL, RDF, SPARQL et les raisonneurs OWL DL comme Pellet sont de plus en plus utilisés dans les nouveaux systèmes basés sur le web sémantique. Ces langages offrent une grande expressivité pour représenter des connaissances, par exemple pour définir en une seule classe le contexte d'un patron d'annotation. Ils permettent d'utiliser directement des raisonneurs déjà programmés et optimisé pour effectuer des inférences complexes à programmer.

Avec OWL 1 DL ou les sous-langages de OWL correspondant à la logique des descriptions, la complétude et la correction des raisonnements sont assurés, c'est à dire que toutes les déductions sont effectuées et qu'aucune déduction effectuée n'est fausse. La difficulté cependant est de bien représenter le domaine en l'absence de l'hypothèse de nom unique et de l'hypothèse du monde clos, caractéristique de la logique des descriptions. Formellement, cela signifie que peu de connaissances sont implicites, ce qui implique concrètement d'expliciter au maximum les exclusions entre classes et instances, l'exhaustivité des sous-classes, etc. Dans ces conditions, la vérification de la cohérence des instances avec l'ontologie est directement assurée par les raisonneurs OWL décidables comme Pellet.

La complexité théorique des calculs et leur efficacité pratique sont difficiles à établir. Les quelques benchmarks disponibles (Bock et al. 2008) ne permettent pas d'extrapoler le temps calcul et des expériences sur de grandes masses de données d'annotation sont encore nécessaires.

Les fonctionnalités rendues possible par les raisonnements automatiques offrent une nette plus value à MemoNote, en augmentant son utilité (recherches sémantiques) et son utilisabilité (activation automatique de patrons).

7 Conclusion et perspectives

L'intégration de raisonnements automatiques dans les Environnements Informatiques pour l'apprentissage Humain basés sur des ontologies et en particulier de l'outil d'annotation MemoNote est jusqu'à présent très limitée. Cela est principalement dû aux propriétés des langages de représentations ontologiques utilisées, qui nécessitent des programmes de

raisonnement ad hoc dont les qualités de décidabilité, d'expressivité et de performance ne sont pas bien définies.

L'utilisation de langages standards du web sémantique pour représenter les ontologies offre non seulement une interopérabilité aux EIAH développés, mais aussi des qualités garanties d'expressivité, de décidabilité, de correction et complétude des raisonnements, à condition d'utiliser un résolveur standard sur ces langages.

En particulier, les raisonnements étendant la transitivité des compositions usuelles (en composant une propriété avec *is_part_of*) de OWL 1 sont possible en OWL 2 en conservant la décidabilité.

L'utilisation de représentations OWL Full n'est pas non plus un problème, à condition que le raisonneur ne les manipule pas directement, comme c'est le cas pour les patrons d'annotation.

L'utilisation de OWL permet ainsi de profiter de la disponibilité de raisonneurs standards et de reposer sur les optimisations qu'il effectue pour améliorer les performances. MemoNote peut ainsi passer à l'échelle du web sémantique et manipuler une masse importante d'annotations de façon sure, interopérable avec les standards du W3C. MemoNote alors communiquer facilement avec n'importe quel raisonneur OWL, comme nous l'avons fait avec Pellet, s'intégrer à des EIAH utilisant ces standards. En particulier, les ontologies décrivant les différents éléments d'une annotation peuvent être précisément celles utilisées par un EIAH particulier. Les raisonnements de MemoNote fonctionnent nativement sur ces ontologies, sans modification du logiciel.

Les nouveaux standards utilisés ont abouti à l'implantation de nouvelles fonctionnalités pour l'enseignant. Ces fonctionnalités répondent à des besoins essentiels pour les enseignants qui utilisent MemoNote, comme la recherche contextuelle sémantique ou l'activation automatique de patron selon le contexte de la séance.

Les questions qui restent à développer concernent principalement la complexité, qui n'est pas bien établie. Les benchmarks des raisonneurs (Bock et al. 2008) permettent de choisir le raisonneur correspondant aux requêtes nécessaires et ayant les meilleures performances. Mais ils ne fournissent pas de résultats théoriques de complexité. Il sera nécessaire donc d'étudier leurs performances sur nos requêtes et nos données, avec un volume important d'annotations.

Le stockage et la manipulation des ontologies et leurs instances (annotations) dans des fichiers est la source de problèmes importants de performances d'accès et de temps de réponse. En effet, avec une grande masse de données, le traitement en mémoire vive tel qu'il est effectué actuellement ne sera pas possible. D'un autre côté, l'utilisation sur le WEB nécessite d'assurer les conditions ACID (Atomicité, Cohérence, Isolation et Durabilité) aux transactions en parallèle des utilisateurs. Ces deux points plaident pour l'intégration de MemoNote avec des systèmes

de gestion des bases de données relationnelles comme Oracle Semantic Store[1]ou Virtuoso (Erling & Mikhailov, 2007) qui offrent un stockage, des accès et des raisonnements efficaces sur des représentations persistantes RDF et OWL tout en assurant les propriétés ACID.

Enfin des besoins plus poussés des enseignants peuvent demander des représentations et des requêtes impossibles avec OWL 2.0 et Pellet. Cela nécessitera dans ce cas d'étudier comment conserver les propriétés attendues avec des programmes propres à ces besoins.

Références

Azouaou F. & Desmoulins C. (2006). MemoNote, an annotation-based personal knowledge management tool for teacher. In Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA). Barcelona, Spain. IADIS Press, pp. 251-258. ISBN: 972-8924-22-4

Bechhofer S., Liebig T. Luther M. Noppens O. Patel-Schneider P. Suntisrivaraporn B. Turhan A. Weithoner T. (2006). DIG 2.0 Towards a flexible interfacefor Description Logic reasoners. In Proceedings of the OWL Experiences and Directions Workshop.

Berners-Lee T.(1998) Semantic Web Roadmap. World Wide Web Consortium (W3C), http://w3.org/DesignIssues/Semantic.html

Biezunski M. Martin B. Steven R. (1998). Topic Maps: Information Technology Document Description and Markup languages, 1999. http://www.y12.doe.gov/sgml/sc34/document/0129.pdf

Ciravegna F. Dingli A. Petrelli D. Wilks Y. (2002). Timely and Non-Intrusive Active Document Annotation via Adaptive Information Extraction, in Semantic Au-thoring, Annotation & Knowledge Markup (SAAKM 2002), ECAI 2002 Workshop, Lyon, France.

Decker S. et al. (2005). TRIPLE - an RDF Rule Language with Context and Use Cases. In W3CWorkshop on Rule Languages for Interoperability, Washington D.C., USA.

Dicheva, D. & Dichev C. (2006). TM4L: Creating and Browsing Educational Topic Maps, British Journal of Educational Technology - BJET (2006) 37(3): 391-404

Bock J. Haase P. Ji Q. Volz R. (2008). Benchmarking OWL reasoners. In Proceedings of the Workshop on Advancing Reasoning on the Web:S calability and Commonsense (ARea08), CEUR-WS. Ã- p. 119

Erling O. & Mikhailov I. (2007). RDF Support in the Virtuoso DBMS. In Proceddings of the 1st Conference on Social Semantic Web, Leipzig, Germany.

Golbreich C. & Wallace E.K. (2009). OWL 2 Web Ontology Language: New Features and Rationale. W3C working draft, W3C http://www.w3.org/TR/2009/WD-owl2-new-features-20090611/.

[1] http://www.oracle.com/technetwork/database/options/semantic-tech/index.html

Henze N. Dolog, P. Nejdl W. (2004). Reasoning and ontologies for personalized e-learning in the Semantic Web. Educational Technology & Society, 7, 4, 82–97.

Horrocks I. Patel-Schneider, P.F. Boley H. Tabet S. Grosof B. Dean M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. Available from http://www.w3.org/Submission/2004/SUBMSWRL-20040521/

Huang W. Webster D., Wood D. Ishaya T. (2006).An intelligent semantic e-learning framework using context-aware Semantic Web technologies. British Journal of Educational Technology, 37(3), 351–373.

iMarkup-Solutions-Inc. iMarkup client Vista, California, USA. Année d'accès 2012. Url du document http://www.imarkup.com

KAHAN J. KOIVUNEN M.-R., PRUD'HOMMEAUX E. SWICK R. (2001) "Annotea: An Open RDF Infrastructure for Shared Web Annotations". In The TenthInternational World Wide Web Conference, Hong Kong, May, pp. 623-632 http://www10.org/cdrom/papers/488/index.html

Kifer M. & G. Lausen G. (1995). Logical foundations of object-oriented and frame-based systems.Journal of the ACM.

Lassila, O. & Swick R. (1999) Resource Description Framework (RDF) Model and Syntax Specification. W3C Proposed Recommendation. http://www.w3.org/TR/PR-rdf-syntax.

Levy M.A. M. O'Connor J. Rubin D. L. (2009). Semantic reasoning with image annotations for tumor assessment," in Proceedings of the American Medical Informatics Association Annual Symposium, vol. 11, pp. 359–363.

McBride B. (2007). An Introduction to RDF and the Jena RDF API. http://jena.sourceforge.net/tutorial/RDF_API/index.html

McGuinness D.L. & van Harmelen F. (2004). OWL Web Ontology Language Overview, World Wide Web Consortium (W3C) recommendation. http://www.w3.org/TR/owl-features.

Noy N.F., et al. (2003). Protégé-2000 : An open-source ontology-development and knowledge acquisition environment, AMIA Annual Symposium Proceedings p. 953.

Prud'hommeaux E. & Seaborn A. (2008). SPARQL Query Language for RDF. W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/

Sirin E. B. Parsia B. Cuenca Grau B. Kalyanpur A. Katz Y. (2003).Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics.

Sperber D. (2001). L'individuel sous influence du collectif, in La Recherche, vol. 344. pp. 32-35.