Aide à la formulation de requêtes dans un médiateur

Help to Queries Expression in a Mediator Approch

C. Reynaud[1,2] B. Safar[1]

[1] L.R.I.- C.N.R.S - Université Paris-Sud
[2] Université Paris X

L.R.I., Bâtiment 490, 91405, Orsay Cedex, France
{cr, safar}@lri.fr

Résumé

La recherche d'informations au sein de sources multiples et hétérogènes est complexe. Les approches médiateurs proposent des solutions qui exploitent une ontologie du domaine. Dans cet article, nous décrivons les apports d'une telle ontologie pour aider à exprimer des requêtes au sein de serveurs d'information et nous détaillons les mécanismes d'inférence mis en œuvre pour atteindre cet objectif. Le travail présenté a été effectué dans le cadre du projet PICSEL[1]*. Les exemples cités sont issus du domaine des produits du tourisme, domaine d'expérimentation choisi dans le projet.*

Mots Clefs

Assistance "intelligente", formulation de requêtes, ontologie, logique de description.

Abstract

Information retrieval over multiple and heterogeneous sources is complex. Mediator approaches propose solutions exploiting a domain ontology. In this paper, we describe the practical benefits of such an ontoloy supporting guidance when querying information servers, then we focus on the inference capabilities performed in order to provide an automated intelligent assistance. The work presented here was developed within the PICSEL project. The examples are coming from the tourism products domain.

Key words

Intelligent assistance, querying process, ontology, description logics.

1 Introduction

La recherche d'information au sein de sources multiples et hétérogènes est complexe. L'utilisateur peut accéder à des quantités très importantes d'information au sein desquelles il doit essayer d'extraire des éléments de réponse pertinents. De plus, chaque source a sa propre origine, son propre vocabulaire, son propre format de représentation des données. Une telle hétérogénéité est un obstacle supplémentaire. Sans aide particulière, l'utilisateur est obligé de s'adapter aux particularités de chaque source interrogée.

Le problème d'intégration d'information qui se pose peut être résolu de deux façons différentes. La première passe par la construction de bases de données réelles, appelées *entrepôts de données*, regroupant toutes les informations pertinentes auxquelles l'utilisateur désire avoir accès. Les requêtes sont posées directement sur les données de l'entrepôt. La seconde consiste à construire un *médiateur* servant d'interface entre les utilisateurs et les sources, et donnant l'impression d'interroger un système centralisé et homogène. Les données restent ainsi stockées et réparties au niveau de chacune des sources sous une forme qui peut varier d'une source à l'autre et le médiateur regroupe dans un cadre unificateur la description de chacune d'elles.

Dans cet article, nous nous situons dans le cadre du projet PICSEL, qui vise à construire des serveurs d'information, ou médiateurs, au-dessus de sources d'information multiples, distribuées et hétérogènes, mais relatives à un même domaine d'application. Le travail présenté porte sur les techniques utilisées dans le projet pour aider des utilisateurs novices à formuler leurs requêtes. En effet, en fournissant à la fois le vocabulaire du domaine utilisable et un langage de requêtes ayant un fort pouvoir d'expression, l'approche médiateur facilite l'expression de requêtes, mais cette tâche reste malgré tout difficile pour un utilisateur novice. Il est ainsi nécessaire de l'aider à "explorer" les connaissances représentées dans le système de façon à ce qu'il comprenne quels termes sont les plus appropriés pour exprimer ce qu'il recherche. De plus, le langage de re-

1. PICSEL est un projet mené par l'équipe IASI du LRI dans le cadre d'un contrat CNET-CTI n°97 1B 378.

quêtes utilisé dans PICSEL est fondé sur un formalisme logique dont la sémantique formelle doit être rendue accessible à des utilisateurs non avertis.

Les techniques proposées facilitent la formulation de requêtes. Elles consistent à exploiter, de façon combinée, les connaissances correspondant à la description du domaine d'application du serveur, appelée *ontologie du domaine*, et les inférences du formalisme logique de représentation des connaissances. Ce formalisme est CARIN-ALN [8], un formalisme qui combine un langage de règles de type Datalog avec une logique de description. C'est un langage riche qui permet une description fine de l'ontologie du domaine. S'agissant d'un langage formel, il a par ailleurs l'avantage de pouvoir faire l'objet de traitements automatisés.

En plus de présenter le vocabulaire du domaine à l'utilisateur de façon structurée, la mise en œuvre dynamique des inférences sur l'ontologie rend possible l'expression de requêtes personnalisées dont la cohérence globale est garantie. Cette aide, générée dynamiquement, est personnalisée dans la mesure où elle est complètement dirigée par la requête ou les éléments de requête que l'utilisateur cherche à décrire de façon plus précise. Elle est également contextuelle car elle reflète à tout moment l'état courant de la requête, résultat de la saisie de l'utilisateur et de connaissances automatiquement déduites par le système. Enfin, dernier aspect essentiel, l'aide proposée rend totalement transparent le processus d'écriture des requêtes dans le langage du médiateur : celles-ci sont générées automatiquement.

Ces différents apports reposent sur la mise en œuvre de mécanismes d'inférence associés au formalisme de représentation des connaissances dans le médiateur. Ces mécanismes ne sont pas nouveaux. Le caractère novateur de notre travail réside dans le fait de les avoir appliqués à la représentation d'une ontologie du domaine dans un contexte de recherche d'information et, plus précisément, d'aide à la formulation de requêtes. Les mécanismes mis en œuvre permettent l'obtention d'une aide “intelligente”.

Le plan de l'article est le suivant. L'ontologie du domaine d'application est présentée en section 2 et le mode d'expression des requêtes dans PICSEL, en section 3. La section 4 décrit les apports d'une ontologie représentée en CARIN-ALN pour aider un utilisateur à formuler sa requête et la section 5, les mécanismes d'inférence utilisés. Les extensions prévues au travail actuellement réalisé sont présentées en section 6 et quelques travaux proches, en section 7. Enfin, nous concluons en section 8.

Les exemples cités sont issus du domaine des produits du tourisme, domaine d'expérimentation choisi au sein du projet PICSEL.

2 L'ontologie du domaine

Dans PICSEL, les connaissances du domaine sont principalement vues comme le support de l'interface entre le système et ses utilisateurs. Nous décrivons ci-dessous (section 2.1) ces connaissances appliquées au domaine des produits du tourisme. Leur représentation s'effectue, dans PICSEL, de manière déclarative au travers de deux composantes : une composante terminologique et une composante déductive, présentées, respectivement, en section 2.2 et 2.3. Le langage hybride CARIN-ALN est un langage riche qui garantit l'obtention d'un système efficace. En effet, le choix de la logique de description ALN assure la décidabilité de la réponse aux requêtes et des raisonnements corrects et complets. Du point de vue représentation, les restrictions dues à l'absence de certains constructeurs dans cette logique sont compensées par l'utilisation de règles et de contraintes.

2.1 Contenu de l'ontologie du domaine des produits du tourisme

Une ontologie spécifique au domaine des produits du tourisme a été construite [9]. L'objectif était de représenter tous les concepts nécessaires aux utilisateurs du médiateur, que ce soient les administrateurs qui décrivent le contenu des sources ou les usagers à la recherche d'informations dans le domaine. Nous avons estimé les besoins des utilisateurs en simulant nous-mêmes leur tâche. A ce jour, l'ontologie construite comprend plus de 200 concepts et 300 propriétés, appelées rôles [2].

Les concepts sont organisés au sein de hiérarchies sans concepts isolés. Une hiérarchie principale a pour racine le concept *Produit*, qui représente ce qui peut se vendre dans le domaine et regroupe les logements, trajets, locations de véhicules, stages. Des hiérarchies secondaires disjointes existent. Elles décrivent des catégorisations d'objets de sous-domaines du domaine d'application (lieu, loisir, prestation, service, équipement). Chaque concept est défini au travers de ses relations avec d'autres concepts. Pour un concept donné, on précise le concept qui le généralise, éventuellement ses propriétés spécifiques ou bien l'ensemble des propriétés nécessaires et suffisantes d'un élément pour appartenir à ce concept.

Les propriétés sont de deux types :

• les propriétés qu'un utilisateur du serveur souhaiterait préciser lors de l'expression d'une requête.

Exemple : une chambre d'hôtel avec deux lits une place et située dans un hôtel avec piscine.

• Les propriétés nécessaires pour structurer et articuler les concepts entre eux. Ainsi la propriété *Lieu-*

2. Dans tout le reste de l'article, la définition des termes *concept* et *rôle* est propre à la logique de description. Un concept représente une classe comme un ensemble d'individus. Un rôle est une relation binaire utilisée pour spécifier leurs propriétés.

DeRésidenceAssocie permet de relier un *Logement* (ex : *ChambreHôtel*) à un *LieuDeRésidence* (ex. : *Hôtel*). Ce lien est important car les critères de choix d'un logement d'un usager peuvent porter sur les caractéristiques du lieu de résidence (adresse, catégorie, etc.) où le logement est situé.

Cette ontologie, bien qu'encore incomplète, est très riche et comprend une grande partie du vocabulaire du domaine des produits du tourisme, concepts élémentaires et relations associées, de façon à rendre possible l'expression de toute requête. Elle comprend de plus différentes sortes de contraintes, pour expliciter les éventuelles incompatibilités entre concepts ou relations, pour traduire les contraintes sémantiques du domaine d'application, ou pour préciser le type de concept sur lequel porte une relation, (ex : le rôle *LieuDeRésidenceAssocie* ne relie des éléments de type *Logement* qu'à des éléments de type *LieuDeRésidence*).

2.2 La composante terminologique

La composante terminologique de CARIN-ALN comprend des définitions et des inclusions de concepts.

- Une définition de concept $C :=$ *ConceptExpression* associe un nom de concept C à une expression de concept de la logique de description ALN. Les concepts de base d'une terminologie sont ceux qui n'apparaissent pas en partie gauche d'une définition.
- Une inclusion de concept $C_1 \sqsubseteq C_2$ est une assertion exprimant le fait que le concept C_1 est inclus dans (on dit aussi "subsumé" par) le concept C_2. La vision logique d'une inclusion $C_1 \sqsubseteq C_2$ est l'implication logique $\forall x (C_1(x) \subseteq C_2(x))$. Comme inclusion de concepts, on autorise les inclusions de la forme :

- $A \sqsubseteq$ *ConceptExpression*, où A est un concept de base,
- $A_1 \sqcap A_2 \sqsubseteq \bot$, où A_1 et A_2 sont des concepts de base.

La logique de description ALN contient les constructeurs de conjonction ($C_1 \sqcap C_2$), de restriction de concept (ALL R C) qui représente l'ensemble des éléments qui ne sont en relation par le rôle R qu'avec des éléments du concept C, les restrictions de cardinalité (ATLEAST n R), (respectivement (ATMOST n R)), qui représentent l'ensemble des éléments qui sont en relation par R avec au moins, (respectivement au plus), n éléments distincts, et la négation $\neg$ (sur les concepts de base uniquement).

La composante terminologique du langage permet la représentation des concepts et des rôles de l'ontologie. Les concepts sont représentés par des relations unaires et les rôles, par des relations binaires. Les exemples ci-dessous en sont une illustration. L'exemple 1 présente la définition du concept *Produit* : un produit a un seul prix, une seule date de début, éventuellement des services ou prestations associés. L'exemple 2 représente la définition du concept *ActivitéSportive* : une activité sportive est une activité dont la nature associée est qualifiée de loisir sportif. *Produit* et *ActivitéSportive* sont tous deux des concepts dits définis en logique terminologique. L'exemple 3 illustre l'inclusion du concept de base (non explicitement défini) *EquipementCulturel* dans le concept *Equipement* et l'exemple 4, la disjonction entre les concepts de base *EquipementCulturel* et *EquipementSportif*.

Exemple 1 :

```
(Produit  :=
   (and (ATLEAST 1 prixAssocié) (ATMOST 1 prixAssocié)
          (ALL prixAssocie  Nombre)
        (ATLEAST 1 dateDébutAssocié)
          (ATMOST 1 dateDébutAssocié)
          (ALL dateDébutAssocié  Date)
        (ALL serviceProduitAssocié  Service)
        (ALL prestationProduitAssocié  Prestation)))
```

Exemple 2 :

(ActivitéSportive :=
(Activité $\sqcap$ (ALL natureActivitéAssocié LoisirSportif)))

Exemple 3 : (EquipementCulturel $\sqsubseteq$ Equipement)

Exemple 4 :

(EquipementCulturel $\sqcap$ EquipementSportif $\sqsubseteq \bot$)

2.3 La composante déductive

La composante déductive de CARIN-ALN comprend :

- Un ensemble de règles : $q(\bar{Y}) \leftarrow p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n)$ permettant de définir des prédicats ordinaires q, d'arité quelconque et n'intervenant pas dans la composante terminologique, en fonctions d'autres relations p_i quelconques (concept, rôle ou prédicat ordinaire). Toutes les variables de $\bar{Y}$ apparaissent dans $\bar{X}_1 \cup \ldots \cup \bar{X}_n$ et les variables des $\bar{X}_i$ distinctes des variables de $\bar{Y}$ sont considérées comme quantifiées existentiellement. Toutes les variables de noms différents sont considérées comme distinctes.
- Un ensemble de contraintes d'intégrités : $p_1(\bar{X}_1) \wedge \ldots \wedge p_m(\bar{X}_m) \rightarrow \bot$, où les p_i sont des noms de relations quelconques.

L'emploi de règles déductives est indispensable pour traduire des relations sur les concepts autres qu'unaire ou binaire (voir l'exemple 5, où l'on définit un Vol Aller et Retour par la relation de nom *VolAR* et d'arité 4, à partir d'un certain nombre de relations de base), pour construire des définitions disjonctives ou pour exprimer les relations inverses de relations définies dans la composante terminologique.

Exemple 5 :

VolAR(villeDépart, dateDépart1, villeArrivée, dateDépart2)
$\leftarrow$ Vol(v1), lieuDépart(v1, villeDépart),
lieuArrivée(v1, villeArrivée), dateDépart(v1, dateDépart1),
Vol(v2), lieuDépart(v2, villeArrivée),
lieuArrivée(v2, villeDépart), dateDépart(v2, dateDépart2),
dateAntérieure(dateDépart1, dateDépart2).

Les contraintes d'intégrité permettent d'exprimer des contraintes sur d'autres prédicats que les concepts de base de la terminologie.

3 L'expression des requêtes dans Picsel

Le médiateur PICSEL permet à un utilisateur d'effectuer des recherches globales, c'est-à-dire en termes du vocabulaire du domaine d'application global du serveur, vocabulaire regroupé au sein de l'ontologie. Les requêtes sont représentées dans PICSEL à l'aide de la composante déductive du langage CARIN. Une requête peut, en effet, être vue comme une règle dont la partie condition est le corps de la requête et dont la conclusion est la tête de la requête :

$$Q(\bar{X}) \leftarrow p_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge p_n(\bar{X}_n, \bar{Y}_n).$$

Les variables $\bar{X} = \bar{X}_1 \cup \ldots \cup \bar{X}_n$ sont appelées *variables distinguées* de la requête et représentent les variables dont l'utilisateur veut obtenir des instances quand il pose la requête. Les autres variables ne servent qu'à exprimer des contraintes sur les variables distinguées et doivent être considérées logiquement comme quantifiées existentiellement. Ainsi la requête de l'exemple 6 exprime la recherche d'instances de chambres telles que ces chambres soient dans un hôtel situé dans un lieu au soleil et équipé d'un golf.

Exemple 6 :

Q(x) ←Chambre(x), LieuDeRésidenceAssocié(x, y), Hôtel(y),
SituéDans(y, l), LieuAuSoleil(l),
EquipementAssocié(y, z), Golf(z).

Les requêtes soumises au médiateur sont traitées par le moteur de requêtes. L'algorithme, implémenté en Java, est décrit dans [6]. A partir d'une requête exprimée sous la forme d'une conjonction d'atomes (concepts, rôles et prédicats ordinaires), le moteur alterne des phases de réécriture de chaque prédicat ordinaire et de chaque concept, avec des phases de vérification de contraintes. La réécriture d'un prédicat ordinaire se fait en remplaçant ce prédicat par le corps des règles qui le définissent. La réécriture d'un concept terminologique consiste à le remplacer par sa définition. Ces phases de réécriture s'effectuent jusqu'à atteindre des regroupements d'atomes correspondant à des descriptions de sources d'informations.

4 Formulation guidée de la requête

L'ontologie met à la disposition des différents utilisateurs (administrateur de sources d'information ou usager posant une requête) tout le vocabulaire de base d'un domaine d'application, c'est-à-dire les termes correspondant aux concepts élémentaires et aux relations associées. A partir de ce vocabulaire, le langage CARIN-ALN permet à chaque utilisateur de traduire toutes sortes d'expressions ou de requêtes en combinant les composants de base, et/ou éventuellement, de redéfinir, par le biais de la partie déductive du langage, sa propre organisation des connaissances suivant ses besoins et son point de vue.

L'ontologie étant riche et très détaillée, cette liberté d'écriture peut être une source de difficultés pour les usagers qui n'effectuent une requête que ponctuellement et risquent de se perdre devant toutes les possibilités.

L'aide à la formulation de requêtes proposée ici s'adresse à ces utilisateurs ponctuels. Elle consiste, d'une part, à les guider dans l'identification des termes pertinents pour la construction des requêtes, et d'autre part, à les soulager des difficultés liées au formalisme du langage. L'ontologie sert de support à l'ensemble des mécanismes d'assistance mis en œuvre.

4.1 Un guide pour le choix des termes

Le premier objectif est d'éviter à l'utilisateur d'avoir à explorer tout le vocabulaire du domaine pour identifier à la fois, les concepts traduisant au mieux sa demande et les relations associées qu'il doit expliciter pour articuler ces concepts entre eux et construire une requête cohérente.

Ce guidage est effectué par proposition de *requêtes générales prédéfinies* adaptées aux recherches les plus courantes des utilisateurs et que ceux-ci peuvent spécialiser par la suite, suivant leurs besoins.

Pour construire ces requêtes générales, nous avons essayé d'identifier les sujets des requêtes les plus souvent exprimées par les utilisateurs effectuant des recherches sur le domaine d'application. Chaque grand sujet ou *thème* est une nouvelle relation, dont la définition est explicitée à partir des composants de base de l'ontologie, à l'aide de la partie déductive du langage.

Remarquons que ces thèmes, (par exemple *HébergementSeul, Séjour, SéjourNeige, SéjourAuSoleil, SéjourEnFamille*, etc.), ne correspondent pas aux sous-concepts de la racine *Produit* de la hiérarchie, mais à des combinaisons de produits de tourisme élémentaires, définissant des concepts "complexes". Par exemple, un *SéjourAuSoleil* correspond à un *Logement* auquel on associe un *LieuDeRésidence* qu'on sait devoir être situé dans un *LieuAuSoleil*. Le concept *CombinéSéjour* permet de plus d'introduire des produits supplémentaires souvent demandés, comme un *Trajet* et/ou la possibilité de pratiquer une *Activité* particulière.

Chaque thème, représenté en CARIN par une relation, est défini par une règle (cf. exemple 7) dont le corps est une conjonction de concepts et de rôles de la partie terminologique et, éventuellement, de prédicats ordinaires définis par des règles.

Exemple 7 :

SéjourAuSoleil(s, p) ← **CombinéSéjour(s)**,
LogementAssocié(s, l), **Logement(l)**,
LieuRésidenceAssocié(l, r), **LieuDeRésidence(r)**,
SituéDans(r, p), **LieuAuSoleil(p)**

CombinéSéjour, Logement, LieuDeRésidence, et *LieuAuSoleil* sont des concepts, *LogementAssocié, LieuRésidenceAssocié* et *SituéDans* sont des rôles.

Les prédicats correspondant aux têtes des requêtes prédéfinies peuvent ainsi être présentés à l'utilisateur pour qu'il commence par choisir le thème recherché (cf. figure 1). Dès qu'un thème est sélectionné, le corps de

HébergementSeul
SéjourCombiné
SéjourNeige
SéjourAuSoleil
SéjourEn Famille
...

FIG. 1 – *La sélection d'un thème*

la règle définissant le prédicat correspondant peut être exploité. Ce corps de règle qui constitue en lui-même une requête cohérente, bien que très générale, contient a priori les principaux termes du domaine pertinents pour l'utilisateur sans que celui-ci ait eu à les chercher ni surtout à les articuler entre eux. Le problème est maintenant d'adapter cette requête générale aux besoins précis de l'utilisateur.

4.2 Un guide pour affiner les termes

Une fois le thème choisi, les différents concepts intervenant dans la requête générale associée, ainsi que toutes leurs spécialisations, sont présentés à l'utilisateur. Ainsi, si la requête prédéfinie sélectionnée est

Sejour	logement	lieuDeResidence	lieuAuSoleil
combineTrajet	location	individuel	AuSoleilEnEte
combineActivite	maison	hotelGite	BassinMéditérannée
séjourLinguistique	appartement	Hotel	AuSoleilToutesSaisons
circuit	chambre	gite	Caraibes
circuitVoiture	chambreHotel	club	Afrique
croisiere	chambreDHotes	bateau	Océanie

FIG. 2 – *Concepts du thème SéjourAuSoleil*

celle définissant le thème *SéjourAuSoleil*, il est possible de la raffiner en spécialisant individuellement et/ou conjointement chacun des concepts intervenant dans sa définition : *CombinéSéjour, Logement, LieuDeRésidence, LieuAuSoleil* (cf. figure 2 qui présente les différentes spécialisations de chaque concept, classées par niveau hiérarchique). Ainsi, dans la figure 3, *Logement* est spécialisé en *ChambreHôtel*. Le raffinement peut également

logement
location
maison
appartement
chambre
chambreHotel
chambreDHotes
cabineBateau

FIG. 3 – *Choix d'une spécialisation de Logement*

s'effectuer par l'introduction de nouvelles propriétés portant sur les concepts intervenant dans la requête.
Toutes les propriétés associées à chacun des concepts de la requête sont accessibles et visualisables ainsi que les concepts auxquels elles sont reliées, et peuvent être directement introduites dans la requête. Ainsi, parmi

Propriétés de ChambreHotel	Concepts associés
LieuRésidenceAssocié	Hôtel
NbreLitsUnePlaceAssocié	Nbre
NbreLitsDeuxPlacesAssocié	Nbre
NbrePlacesAssocié	Nbre
PetitDejInclusAssocie	ouiNon
PrestationsChambreAssociées	PrestationChambre
SanitaireAssocie	Sanitaire
ClimatisationAssocié	ouiNon

FIG. 4 – *Visualisation des propriétés associées au concept ChambreHôtel*

les propriétés associées au concept *ChambreHôtel*, (cf. figure 4), on peut souhaiter expliciter des prestations particulières pour la chambre par l'intermédiaire du rôle *PrestationChambreAssocié*, et, après avoir visualisé les spécialisations du concept relié, *PrestationChambre*, (cf. figure 5), on peut choisir une *Terrasse*.

PrestationChambre
TV
Terrasse
Coffre-fort

FIG. 5 – *Raffinement du concept PrestationChambre*

L'utilisateur peut réitérer ce processus aussi longtemps qu'il le souhaite. Il pourrait, par exemple, demander à consulter les propriétés de *Terrasse*, et si celles-ci sont associées à des concepts pouvant être spécialisés, il peut poursuivre l'affinement de sa requête. Il pourrait aussi conjointement préciser à ce stade le type de sanitaire souhaité dans la chambre. A chaque étape, l'utilisateur peut, soit simplement visualiser, soit effectivement sélectionner une ou plusieurs options.

4.3 Génération automatique des requêtes

L'aide apportée aux utilisateurs permet de les décharger totalement de la connaissance du langage du médiateur. Les requêtes raffinées par l'utilisateur, sont en effet générées automatiquement en CARIN-ALN par le système, qui intègre l'ensemble des choix effectués en mettant à jour, au fur et à mesure, la requête prédéfinie correspondant au thème initial choisi.
Ainsi, la spécialisation d'un concept (cf. figure 3) conduit à remplacer dans la requête, le concept initial général (*Logement* dans l'exemple 7) par le concept plus spécialisé (*ChambreHôtel* dans l'exemple 7bis).

Exemple 7bis :
SéjourAuSoleil(s, p) ← CombinéSéjour(s),
LogementAssocié(s, l), **ChambreHôtel(l)**,
LieuRésidenceAssocié(l, r), **Hotel(r)**,
SituéDans(r, p), LieuAuSoleil(p)

La sélection d'une propriété (cf. figures 4 et 5) entraîne son ajout dans la requête comme le montre l'exemple 7ter. Ici, le choix de *Terrasse* comme concept lié au concept *ChambreHôtel* par le rôle *PrestationChambreAsso-*

cié conduit à ajouter les atômes *PrestationChambreAssocié*(l, c) et *Terrasse*(c) à la requête initiale :

Exemple 7ter :

SéjourAuSoleil(s, p) ← CombinéSéjour(s),
LogementAssocié(s, l), ChambreHôtel(l),
PrestationChambreAssocié(l, c), Terrasse(c)
LieuRésidenceAssocié(l, r), Hotel(r),
SituéDans(r, p), LieuAuSoleil(p)

Notons, sur l'exemple 7ter, que le système génère automatiquement les variables utiles pour l'articulation des différentes relations introduites et gère les liaisons entre ces variables pour garder une requête cohérente. Ainsi, le rôle *PrestationChambreAssocié*, représenté par une relation binaire, est introduit avec deux variables, l'une correspondant à la variable déjà présente dans la requête qui référence le concept *ChambreHôtel*, l'autre nouvellement créée, pour référencer le concept *Terrasse*.
L'assistance fournie, la présentation des spécialisations d'un concept et de toutes ses propriétés associées, est dirigée au départ par le thème choisi et les concepts faisant partie de sa définition. L'assistance suit ensuite les choix de l'utilisateur et s'adapte au fur et à mesure en fonction des raffinements effectués. C'est en ce sens que l'aide que procure l'ontologie CARIN-ALN peut être qualifiée d'aide personnalisée. Sa réalisation est présentée ci-dessous.

5 Mécanismes d'inférence mis en œuvre

L'assistance à la formulation de requêtes est basée sur deux fonctionnalités principales :

- la possibilité de raffiner des concepts, qui exploite les liens de spécialisation entre concepts,
- la possibilité d'ajouter des propriétés de concepts, qui exploite les définitions de concepts et la notion d'héritage de propriétés.

A partir des concepts intervenant dans une requête prédéfinie, l'utilisation alternée de ces deux types de fonctionnalités permet un parcours de l'ontologie à la fois en profondeur et de façon latérale. L'utilisateur peut ainsi accéder à tous les éléments pertinents permettant de préciser le thème choisi, et ce directement, sans avoir à naviguer dans l'ontologie pour rechercher les concepts et les propriétés qu'il souhaite préciser, et surtout sans avoir à se préoccuper de la syntaxe du langage, comme de l'aspect liaison de variables.
Les deux types de fonctionnalités exploitent des mécanismes spécifiques au langage de représentation de l'ontologie, un langage formel autorisant des traitements automatiques. Nous décrivons ces mécanismes dans les sections qui suivent.

5.1 Liens de généralisation/spécialisation

Le raffinement de concepts (cf. figures 3 ou 5) est basé sur le fait que l'ontologie ne contient pas de concepts isolés, et donc que chaque concept appartient toujours à une hiérarchie. Cela signifie qu'un concept peut être soit spécialisé, soit lui-même la spécialisation d'un ou plusieurs concepts plus généraux. En se basant sur ces liens de spécialisation, il est ainsi possible de faire état de tous les fils de chaque concept apparaissant dans la requête prédéfinie associée à un thème sélectionné. Pour les utilisateurs, cela permet de préciser le type des thèmes recherchés.

5.2 Propriétés des concepts et notion d'héritage

L'introduction de nouvelles propriétés dans une requête (cf. figure 4) est basée sur la présentation de toutes les propriétés associées aux concepts intervenant dans la requête en cours de construction. Ces propriétés concernent à la fois les propriétés spécifiques à un concept, c'est-à-dire qui interviennent directement dans sa définition, et celles héritées des concepts plus généraux. Dans l'exemple de la figure 4, la propriété *LieuRésidenceAssocié* est une propriété héritée du concept *Logement*, la restriction aux hôtels est spécifique au concept *ChambreHôtel* et toutes les autres propriétés sont héritées du concept père *Chambre*.
Lors de la représentation de la terminologie, les rôles ont été définis en fonction des concepts qu'ils lient alors qu'en logique de description, un rôle peut normalement être utilisé pour relier des concepts différents. Ainsi, le même rôle *PrestationAssocié* aurait pu être utilisé comme propriété commune aux concepts *LieuDeRésidence* et *Chambre*. Nous avons fait le choix d'utiliser des noms de rôles identiques lorsque les concepts concernés étaient liés par des liens de généralisation /spécialisation et des noms distincts dans le cas contraire. Ainsi, puisque *Chambre* est une spécialisation de *Logement*, nous pouvons associer à ces deux concepts le rôle commun *LieuDeRésidenceAssocié*. En revanche, comme *Chambre* et *LieuDeRésidence* n'appartiennent pas à la même hiérarchie, nous distinguons les *PrestationChambreAssocié* et les *PrestationLieuDeRésidenceAssocié*. Ces distinctions sont utiles pour l'utilisateur, car les noms des rôles évoquent clairement les concepts caractérisés. Ils deviennent ainsi plus compréhensibles et évitent les ambiguités.

5.3 Contraintes de typage

Pour chaque propriété, le nom du concept avec lequel cette propriété établit un lien, est donné. Cette information peut être représentée dans l'ontologie de deux façons différentes : soit dans la définition du concept auquel est associée la propriété, par le biais d'une restriction de concept (cf. exemple 8, où le rôle *LieuDeRésidenceAssocié* ne relie les éléments de type *ChambreHôtel* qu'à des éléments de type *Hôtel*), soit par une contrainte explicite de typage précisant le domaine de valeurs des arguments de relations et exprimée dans la partie déductive du langage (cf. exemple 9). Quelle que soit la forme, c'est une information qui est systématiquement représentée dans l'ontologie de façon à ce que l'utilisateur puisse mieux comprendre la signification des propriétés.

Exemple 8 :
ChambreHôtel := (and Chambre
(ALL LieuDeRésidenceAssocié Hôtel))

Exemple 9 :
PrestationChambreAssocié(x,y), $\neg$ PrestationChambre(y) $\rightarrow \perp$
PrestationChambreAssocié (x,y), $\neg$ Chambre(x) $\rightarrow \perp$

Le concept avec lequel la propriété décrite établit un lien peut à son tour être spécialisé, s'il a des concepts fils dans l'ontologie et si l'utilisateur le demande. Un processus de raffinement peut alors avoir lieu (cf. fig. 5) et, si le concept a des propriétés, celles-ci pourront à leur tour être affichées et précisées.

5.4 Inférences de propriétés

Dès qu'un concept est sélectionné, les contraintes exprimées dans sa définition, au travers des restrictions de concepts (expressions du type (ALL R C)), sont exploitées. Le mécanisme d'inférence mis en œuvre à ce stade est propre au langage terminologique. Soit la définition de concept, $C := C' \sqcap$ (ALL R A), qui définit C comme une spécialisation du concept C' qui n'est en relation par l'intermédiaire du rôle R qu'avec des concepts de type A. La sélection de C dans la requête déclenche automatiquement la propagation de la restriction de concept qui lui est associé, (ALL R A), et conduit à restreindre dans la requête, toutes les occurences de concepts en relation avec C par le rôle R, à des concepts de type A.
Par exemple, dans la terminologie, le concept *ChambreHôtel* est défini comme une spécialisation du concept *Chambre*, auquel on ne relie, par la propriété *LieuDeRésidenceAssocié*, que des concepts *Hôtel* (cf. exemple 8). On peut noter l'exploitation de cette contrainte dans la figure 4 et l'exemple 7bis. Le choix de spécialiser *Logement* en *ChambreHôtel* conduit à spécialiser *LieuDeRésidence* en *Hôtel*, à la fois dans la fenêtre affichant les propriétés de *ChambreHôtel* et dans la requête spécialisée construite. Cette nouvelle information, traduisant le fait que le lieu de résidence recherché par l'utilisateur doit être un hôtel s'il désire un logement de type *ChambreHôtel*, est déduite automatiquement, sans que l'utilisateur n'ait à le préciser. Elle est prise immédiatement en compte par les modules d'aide, ce qui permet de qualifier l'aide fournie aux utilisateurs de contextuelle car intégrant à tout moment les choix antérieurs de l'utilisateur.

6 Extensions prévues

Une première maquette d'aide à la formulation de requêtes, qui exploite les mécanismes présentés précédemment et qui construit la requête, a été développée en Java. Elle a été construite autour de deux types de fenêtres : des fenêtres de raffinement de concepts exploitant les liens de spécialisation entre concepts et des fenêtres de propriétés de concepts exploitant les définitions des concepts et la notion d'héritage de propriétés. D'autres mécanismes d'aide pouvant augmenter le confort de l'utilisateur ont été étudiés mais ne sont pas encore implémentés.

6.1 Prise en compte de contraintes non terminologiques

D'autres contraintes sont exprimables en CARIN-ALN, à l'aide de la partie déductive du langage. Elles ne sont pas considérées à ce stade, pour ne pas ralentir l'interaction. Par exemple, dans la terminologie, le concept *CabineBateau* est défini comme une spécialisation de *Chambre*. Il hérite, à ce titre, de toutes les propriétés associées à *Chambre*, donc, entre autres, des prestations associées. Toutefois, une contrainte (cf. cidessous) exprime le fait que les terrasses font exception et ne font pas partie des prestations associées à une cabine de bateau.

Exemple 10 :
CabineBateau(l), PrestationChambreAssocié(l, p),
Terrasse(p) $\rightarrow \perp$

La propagation de cette contrainte ne se fait aujourd'hui que dans une phase ultérieure, où la requête complètement définie par l'utilisateur est confrontée en une fois par le moteur de requêtes, à toutes les contraintes exprimées sur le domaine. Ainsi, actuellement, dans le prototype développé, l'utilisateur peut demander à spécialiser *Logement* en *CabineBateau* et demander de plus à ce que cette dernière possède une terrasse. Bien sûr, dès que la requête est transmise au moteur, les contraintes sont propagées et l'inconsistance détectée. La requête est alors transmise à un autre module, le module de réparation de requêtes [2], qui identifie les termes à l'origine de l'inconsistance et suggère des modifications de la requête reçue. Typiquement ici, le module de réparation proposera de re-généraliser l'un ou l'autre des concepts concernés, soit *CabineBateau* en *Chambre* soit *Terrasse* en *Prestation* de façon à éliminer l'inconsistance tout en proposant une requête proche de la requête initiale. Mais il paraîtra frustrant à l'utilisateur de devoir revenir sur les choix qu'il vient juste d'effectuer. Une des perspectives de ce travail est donc d'étudier quelles contraintes supplémentaires, non terminologiques, pourraient être exploitées à ce stade sans trop ralentir la saisie des requêtes.

6.2 Affichage progressif des propriétés

Dans la première maquette, toutes les propriétés associées à un concept dans l'ontologie sont affichées en une fois quand l'utilisateur le souhaite. Le problème est que ces propriétés peuvent être nombreuses, et ne sont pas toujours pertinentes pour l'utilisateur.

En effet, nous avons essayé d'exprimer dans l'ontologie tout le vocabulaire utile sur le domaine que ce soit pour l'usager qui pose une requête mais aussi pour l'administrateur de sources qui décrit ses produits. Or, certaines propriétés sont clairement destinées au deuxième type d'utilisateur plutôt qu'au premier. Par exemple, le concept *LieuDeRésidence* qui regroupe les Hôtels, les Clubs, les Gîtes, etc., est un concept central de l'ontologie auquel on associe les nombreuses caractéristiques partagées par tous les logements (les chambres) qui y sont rattachés. La grande majorité de ces caractéristiques peuvent être choisies par l'usager mais certaines ne concernent que les administrateurs de source qui doivent pouvoir délivrer l'information correspondante (comme le numéro de téléphone du lieu, les heures d'arrivée et de départ théoriques de la résidence, etc.) sans que celle-ci soit un critère de sélection pour l'usager. Ces caractéristiques ne devraient donc pas être visualisables dans l'interface de formulation de requête, ce qui allègerait sensiblement l'affichage.

De plus, parmi les caractéristique interrogeables, certaines sont des critères de choix essentiels pour tous les usagers, comme la localisation, la catégorie ou les équipements sur place. D'autres propriétes traduisent des caractéristiques qui n'intéressent que certains usagers, comme les animations spécifiques enfants, le nombre de chambres accessibles aux handicapés, etc. Il faudrait donc pouvoir ne faire apparaître, dans un premier temps, que les propriétés essentielles pour tous de façon à ne pas noyer immédiatement l'utilisateur sous trop de choix.

Le problème posé ici est double. Premièrement, toutes les propriétés sont aujourd'hui explicitées de la même façon dans l'ontologie. Il nous faut alors chercher comment les distinguer pour que la représentation et le mécanisme de visualisation associé reste simple. Deuxièmement, si certaines propriétés interrogeables par l'utilisateur ne sont pas immédiatement accessibles, il faut trouver une façon de les ré-introduire à la demande, sans alourdir l'interface. La prise en compte d'un modèle de l'utilisateur pourrait être une solution appropriée à ce deuxième problème, qui permettrait de plus de mieux personnaliser l'aide apportée.

6.3 Autres extensions possibles

Le problème de l'affichage progressif se pose aussi avec les spécialisations de concepts. L'affichage se fait actuellement en profondeur d'abord, (toutes les spécialisations du premier fils d'un concept sont présentées avant l'affichage du deuxième fils). Cette façon de faire nous semble avoir l'avantage de permettre à l'utilisateur de mieux appréhender les distinctions entre concepts frères (par la réunion des fils de l'un et l'opposition aux fils de l'autre) et de pouvoir faire directement le choix le plus précis possible s'il est fixé sur une option. En revanche, elle peut avoir l'inconvénient de perdre les utilisateurs plus indécis en ne mettant pas assez en relief les spécialisations intermédiaires et en les orientant vers des concepts trop spécialisés sur lesquels ils seront conduits par la suite à revenir. Ce type de choix n'est pas de notre ressort et ne peut être effectué qu'après une validation conséquente auprès d'utilisateurs potentiels.

7 Travaux proches

Les travaux qui s'intéressent à faciliter l'expression de requêtes sont de différents types.

Certains ont étudié les aspects visualisation et navigation au sein d'une ontologie en exploitant la géométrie hyperbolique [5]. Bien que n'ayant pas été approfondis, ces aspects ont aussi été étudiés dans PICSEL. Un module a été développé. Il permet à l'utilisateur de visualiser la hiérarchie des concepts sous forme d'un graphe à l'aide de l'utilitaire Graphlet, d'avoir accès à la liste des concepts et des rôles de l'ontologie et pour chaque concept, de consulter sa définition, ses généralisants directs, ses spécialisations et les rôles qui lui sont associés. Ce module permet ainsi de naviguer dans l'ontologie et de consulter le vocabulaire disponible. Ceci est très utile pour aider à comprendre la signification de l'ensemble des concepts et rôles représentés et peut guider dans le choix des termes à utiliser pour exprimer une requête. En revanche, lorsque l'ontologie est très détaillée et très riche, comme dans le cas de notre application, l'utilisation d'une telle interface devient difficile pour un utilisateur non averti. Ceci explique notre choix de le guider davantage en lui proposant des thèmes sur lesquels faire porter ses recherches et en lui permettant de les affiner. D'autres travaux se sont intéressés à faciliter l'expression de requêtes en proposant des langages adaptés à la recherche d'informations dans des documents semi-structurés. Dans [1], par exemple, les auteurs ont travaillé à enrichir le pouvoir d'expression de langages d'interrogation de Bases de Données orientées Objet, mais n'ont pas abordé les aspects d'interaction avec l'utilisateur.

Notre travail se rapproche davantage des techniques proposées en recherche d'information qui exploitent une ontologie pour assister l'utilisateur.

Parmi ces techniques figurent celles proposées par les outils de navigation disponibles actuellement sur le Web. Ainsi, Yahoo [14] propose aux utilisateurs d'effectuer une recherche soit par mots-clefs soit en précisant le domaine de la recherche. Il dispose pour cela d'une vue hiérarchique de domaines d'intérêts auxquels sont rattachées des pages web. Cette facilité est intéressante. Néanmoins, les domaines sont très larges. Les connaissances représentées correspondent uniquement à des inclusions de concepts sans qu'aucun concept ne soit défini. Le langage d'expression des requêtes est limité et ne permet que de sélectionner le domaine sur lequel porte la requête de l'utilisateur. Ainsi l'outil ne permet pas d'effectuer des recherches très précises.

Permettre l'expression de requêtes fines et précises est un des objectifs réalisables par les projets exploitant une ontologie représentée dans un langage formel. C'est le cas du projet Untangle développé au départ, en 1994, par Chris Welty [11] pour la recherche d'information sur le Web et qui se poursuit actuellement dans le domaine de la recherche documentaire [13] et, également des travaux de [10] portant sur la conception de portails Web s'adressant à une communauté d'utilisateurs.

Le projet Untangle propose une interface exploitant une ontologie représentée dans la logique de description CLASSIC. L'utilisateur définit sa requête à l'aide d'une interface graphique et n'a pas à saisir l'expression textuelle associée. L'ontologie est présentée sous forme d'une hiérarchie. Une requête utilisateur porte toujours sur un concept, et un seul, que l'utilisateur sélectionne dans la hiérarchie qui lui est présentée. Il peut ensuite préciser sa recherche en spécialisant le concept sélectionné au départ ou en indiquant les propriétés qui doivent être vérifiées. L'interface est personnalisée puisqu'elle est dirigée par le choix du concept sélectionné au départ par l'utilisateur. Nos techniques sont proches de celles mises en oeuvre dans ce projet puisqu'il s'agit dans les deux cas d'exploiter une ontologie et des mécanismes d'inférence associés aux logiques de description. Néanmoins, dans Untangle, l'utilisateur doit obligatoirement exprimer ses requêtes à partir de la hiérarchie de concepts qui lui est présentée.

Dans PICSEL, cette contrainte n'existe pas. La partie terminologique de l'ontologie est une modélisation d'un domaine d'application, rassemblant tous les "termes de base" du domaine d'application du serveur. Il s'agit de descriptions de concepts, de catégorisations d'objets révélatrices d'une organisation particulière des connaissances. Comme cette organisation ne correspond pas forcément à la vision de tous les utilisateurs du serveur (usagers posant des requêtes et administrateurs décrivant le contenu de sources d'information), la partie déductive du langage permet, dans PICSEL, de définir des termes correspondant à d'autres points de vue que celui représenté dans la partie terminologique de l'ontologie. Ainsi le thème *SéjourAuSoleil* n'est pas un concept apparaissant dans la hiérarchie des concepts, mais il a été défini par règle à partir des éléments de celle-ci. De ce fait, l'interface proposée dans PICSEL est plus proche des usagers, plus facile à utiliser (définition des requêtes beaucoup plus rapide) car ceux-ci peuvent y trouver directement le thème correspondant à ce qu'ils recherchent. Ils n'ont pas à le redéfinir lorsque celui-ci ne correspond pas à un terme dans la hiérarchie des concepts, une tâche pas toujours facile lorsque la structure des connaissances n'est pas parfaitement maîtrisée.

Les deux projets se distinguent aussi sur les mécanismes d'inférence utilisés. Untangle n'exploite aujourd'hui que les mécanismes d'inférences concernant les instances (ABox) de la terminologie CLASSIC, offrant ainsi des facilités de saisie des instances dans le système. Dans PICSEL, les mécanismes d'inférence associés au langage CARIN-ALN sont mis en oeuvre sur les concepts de la terminologie : exploitation des contraintes de typage, déduction d'informations par exploitation des restrictions de concepts. Ces mécanismes permettent d'avoir une interface contextuelle, qui reflète à tout moment l'état courant de la requête, allant jusqu'à déduire des connaissances conceptuelles non explicitement exprimées par un utilisateur effectuant une recherche.

Dans [10] appliqué à la conception de portails Web s'adressant à une communauté d'utilisateurs, les auteurs proposent des techniques de navigation et de formulation de requêtes, basées sur l'exploitation d'une ontologie, composée à la fois de taxinomies et de règles permettant des inférences. L'objectif premier est d'éviter aux utilisateurs l'usage du langage formel F-Logic. Ainsi, des liens hypertextes contiennent des requêtes qui sont évaluées dès qu'un utilisateur clique dessus. Par ailleurs, des formulaires permettent de sélectionner le concept sur lequel porte une requête ainsi que les propriétés que les instances recherchées doivent vérifier. Dans ces travaux, les requêtes prédéfinies sont relativement simples. En général, il s'agit de la recherche d'instances de concepts représentés dans l'ontologie mais le langage pourrait permettre l'expression de requêtes plus complexes. Le système est par ailleurs capable de générer des connaissances non explicitement représentées, par exploitation de règles.

Ce travail présente de grandes similarités avec le nôtre : requêtes prédéfinies, structure du langage exploitée, génération automatique des requêtes dans le langage requis, mise en oeuvre de mécanismes d'inférence permettant de déduire des connaissances non explicitement représentées. La différence essentielle distinguant ce travail de celui réalisé dans PICSEL est liée aux propriétés des traitements automatisés qui y sont effectués. La complétude du raisonnement n'est pas garantie sur les connaissances représentées en F-Logic. Ainsi, certaines inférences, qui sont fondées logiquement, peuvent, dans certains cas, ne pas être effectuées. Au contraire, les choix de représentation faits dans PICSEL garantissent la décidabilité de la réponse aux requêtes et des raisonnements corrects et complets. Par ailleurs, ces travaux s'appuient sur l'architecture Ontobroker [3] qui comprend une interface facilitant la formulation de requêtes. Dans cette architecture, les mécanismes d'inférence déduisant de nouvelles connaissances sont utilisés lors du traitement des requêtes mais sont sans influence sur l'interface [5]. Notons enfin l'existence de travaux proches dans le domaine de la représentation des connaissances par objets. Dans [4], par exemple, l'auteur a étudié l'aide pouvant être apportée à un utilisateur désirant interroger une base de connaissances représentée dans un langage objet. L'aide exploite les liens de spécialisation et la notion d'héritage de propriétés. Elle permet aux utilisateurs de construire des requêtes complexes en exploitant la notion de points de vue multiples via des formulaires. Toutefois, contrairement à l'aide proposée dans PICSEL, le système développé offre une aide plus limitée car il n'exploite pas les contraintes de typage ni les inférences de propriétés.

8 Conclusion

Par rapport à l'ensemble de ces travaux, notre approche consiste à exploiter une ontologie du domaine correspondant à la description d'un domaine d'application d'un serveur PICSEL et les apports du formalisme logique CARIN-ALN. L'originalité du travail décrit réside dans l'exploitation de mécanismes d'inférence puissants pour l'aide à la construction de requêtes, dans un contexte garantissant des raisonnements corrects et complets. Une interface mettant en œuvre l'ensemble des mécanismes présentés en section 4 et 5 a été développée en Java (développement à base de servlets Java). Son utilisation dans un cadre expérimental à plus grande échelle est en cours, ce qui devrait permettre prochainement une réelle évaluation. Les perspectives dégagées portent sur la prise en compte de contraintes supplémentaires permettant d'accroître les contrôles de cohérence des requêtes par rapport à l'ontologie ainsi que sur l'affichage progressif des propriétés et des spécialisations de concepts (cf. section 6). Il s'agit également d'étudier si l'aide apportée par les mécanismes mis en œuvre est adaptée aux administrateurs des sources d'information qui doivent saisir la description du contenu de leur source ou si de nouvelles fonctionnalités doivent être envisagées.

Remerciements

Nous remercions sincèrement M.-C. Rousset, responsable du projet PICSEL, pour ses relectures attentives et nos fructueuses discussions.

Références

[1] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte and J. Siméon, Querying Documments in Object Databases In *Journal of Digital Libraries* Vol. 1, n^o1, pp. 5-19, 1997.

[2] A. Bidault, C. Froidevaux and B. Safar, Repairing Queries in a Mediator approach. In *Proc. of the 14th European Conf. on artificial intelligence (ECAI 2000)*, Berlin, pp. 406-410, August 2000.

[3] S. Decker, M. Erdmann, D. Fensel and R. Studer, Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. In *Meersmann et al., editors, Database Semantics: Semantic Issues in Multimedia Systems, Kluwer Academic Publishers*, pp. 351-369, 1999.

[4] J. Euzenat, HYTROPES: a www Front-End to an Object Knowledge Management System. In *Proc. of the 10th Knowledge Acquisition Workshop Demonstration Track*, Banff, Canada, 1996.

[5] D. Fensel, S. Decker, M. Erdmann and R. Studer, Ontobroker: The Very High Idea. In *Proceedings of the 11th Int. Conf. Flairs (FLAIRS-98)*, Sanibal Island, Florida, USA, May 1998.

[6] F. Goasdoue, V. Lattes and M.-C. Rousset, The Use of CARIN Language and Algorithms for Information Integration: the PICSEL Project. In *International Journal of Cooperative Information Systems*, Vol. 9, n^o4, pp. 383-401, 2000.

[7] N. Guarino, C. Masolo and G. Vetere, OntoSeek: Content-Based Access to the Web. In *IEEE Intelligent Systems* , pp. 70-80, May/June 1999.

[8] A. Levy A. and M.-C. Rousset, Combining Horn rules and Description Logics in CARIN. In *Artificial Intelligence*, n^o104, pp. 165-209, 1998.

[9] C. Reynaud, B. Safar and H. Gagliardi, Une expérience de représentation d'une ontologie dans le médiateur PICSEL. In *Plateforme de l'AFIA, Ingéniérie des connaissances, IC'01*, pp. 329-348, Grenoble, Juin 2001.

[10] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnuur, R. Studer and Y. Sure, AI for the Web - Ontology-based Community Web Portals. In *AAAI 2000*. pp. 1034-1039, Austin, Texas, 2000.

[11] C. Welty, Knowledge representation for Intelligent Information Retrieval. In *Proceedings of the CAIA'94 Workshop on Intelligent Access to Digital Libraries*, March 1994.

[12] C. Welty, Intelligent Assistance for Navigating the Web. In *Proceedings of the Florida AI Research Symposium*, 1996.

[13] C. Welty and N. Ide, Using the Right Tools : Enhancing Retrieval from Marked-Up Documents. In *J. Computers and The Humanities*, Vol. 33,n^o10, pp. 59-84, April 99.

[14] http://www.yahoo.com