Information-Interaction-Intelligence, Volume 2, n°1

Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL

M.-C. Rousset, A. Bidault, C. Froidevaux, H. Gagliardi, F. Goasdoué, C. Reynaud et B. Safar

Laboratoire de Recherche en Informatique,
Bâtiment 490, Université Paris-Sud, F-91405, Orsay Cedex, France
{mcr,bidault,chris,gag,fg,cr,safar}@lri.fr

Résumé

Le nombre croissant de données accessibles via des réseaux (intranet, internet, etc.) pose le problème de l'intégration de sources d'information préexistantes, souvent distantes et hétérogènes, afin de faciliter leur interrogation par un large public. Une des premières approches proposées en intégration d'informations prône la construction de *médiateurs*. Un médiateur joue un rôle d'interface de requêtes entre un utilisateur et des sources de données. Il donne à l'utilisateur l'illusion d'interroger un système homogène et centralisé en lui évitant d'avoir à trouver les sources de données pertinentes pour sa requête, de les interroger une à une, et de combiner lui-même les informations obtenues.

L'objectif de cet article est de présenter le projet PICSEL [1] qui offre un environnement déclaratif de construction de médiateurs. PICSEL se distingue des systèmes d'intégration d'informations existants par la possibilité d'exprimer le schéma du médiateur dans un langage (CARIN) combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (la logique de description $\mathcal{ALN}$). PICSEL intègre un module d'affinement de requêtes, première brique d'un module de dialogue coopératif entre un médiateur et ses utilisateurs.

Mots-clés : Médiateur, ontologie, réécriture et affinement de requêtes

1. Le projet PICSEL a été financé par France Telecom R&D dans le cadre d'une CTI

1 INTRODUCTION

Avec l'émergence de l'Internet et de l'Intranet, il est possible d'accéder aujourd'hui à de multiples sources d'informations réparties, hétérogènes et autonomes, qui peuvent contenir des informations pertinentes et complémentaires pour une application. On peut distinguer deux grandes approches pour l'intégration de sources d'information.

L'approche *entrepôts de données* consiste à voir cette intégration comme la construction de bases de données réelles, appelées *entrepôts*, regroupant les informations pertinentes pour les applications considérées. L'utilisateur pose alors ses requêtes ou lance un traitement directement sur les données stockées dans l'entrepôt. Les problèmes posés par la construction d'un entrepôt à partir de plusieurs bases de données spécialisées concernent la définition de son schéma, son peuplement et sa mise à jour, en fonction des différentes sources d'information à partir desquelles il est construit. Les travaux récents sur l'intégration de données semi-structurées (e.g., [12, 17, 13, 33]) ou la migration de données d'un format dans un autre (e.g., [26, 2]) relèvent de cette approche.

L'approche *médiateur* ([32]) consiste à fonder l'intégration d'informations sur l'exploitation de *vues abstraites* décrivant le contenu des différentes sources d'information. Les données ne sont pas stockées au niveau du médiateur et ne sont accessibles qu'au niveau des sources d'information. L'interrogation et la détermination des sources d'information pertinentes nécessitent la construction de *plans de requêtes* dont l'exécution permettra d'obtenir l'ensemble des réponses à partir des sources disponibles. Les deux approches peuvent être combinées, comme dans XYLEME ([33, 34]), dont le but est de construire un entrepôt dynamique regroupant l'ensemble des documents XML du Web. Dans XYLEME, un médiateur, construit au dessus de l'entrepôt, joue un rôle d'interface de requêtes entre les utilisateurs et les documents XML relatifs à un même sujet mais pouvant être sémantiquement hétérogènes ([30, 14]).

Dans cet article, nous nous focalisons sur l'approche médiateur qui présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leur source d'origine. Un médiateur donne à l'utilisateur l'illusion d'interroger un système homogène et centralisé en lui évitant d'avoir à trouver les sources de données pertinentes pour sa requête, de les interroger une à une dans leur langage

spécifique, selon leur schéma particulier et de combiner lui-même les informations obtenues. Le schéma d'une source de données, dit *schéma local*, représente les structures servant de conteneur pour stocker des données dans cette source.
Un utilisateur pose ses requêtes dans les termes du *schéma du médiateur*. Ce schéma, souvent appelé aussi *schéma global*, regroupe l'ensemble des prédicats modélisant le domaine d'application du système. Il joue donc le rôle de l'ontologie du domaine qui fournit un vocabulaire structuré servant de support à l'expression de requêtes et c'est par son intermédiaire que se fait l'intégration.

Le médiateur ne peut pas évaluer directement les requêtes qui lui sont posées car il ne contient pas de données. Ces dernières sont en fait stockées de façon distribuée dans des sources indépendantes. Le médiateur ne dispose que de *vues* abstraites des données stockées dans les sources. L'interrogation effective des sources se fait via des adaptateurs (appelés des *wrappers* en anglais) qui traduisent les requêtes exprimées par les vues dans le langage de requêtes spécifique accepté par chaque source.

Les réponses à une requête posée à un médiateur peuvent être calculées par *réécriture en termes de vues* de cette requête. Le problème consiste alors à trouver une requête qui, selon le choix de conception du médiateur est *équivalente à* ou *implique logiquement* la requête de l'utilisateur mais n'utilise que des vues. Cette requête, nommée *réécriture*, est exprimée *en termes des vues* disponibles dans un langage particulier, appelé *langage de réécritures*. Les réponses à la requête posée sont alors obtenues en évaluant les réécritures de cette requête sur les extensions des vues.

L'architecture générale commune à tous les médiateurs est illustrée dans la figure 1.

Les deux principaux problèmes posés par la construction d'un médiateur sont :

- le choix du langage utilisé pour modéliser le schéma global, ainsi que le choix des langages pour modéliser, en fonction de ce schéma, les vues sur les sources à intégrer et les requêtes des utilisateurs,

- et, en fonction de ces choix de modélisation, la conception et la mise en œuvre d'algorithmes de réécritures de requêtes en termes de vues pour le calcul des plans de requêtes à exécuter afin d'obtenir l'ensemble des réponses à une requête globale.

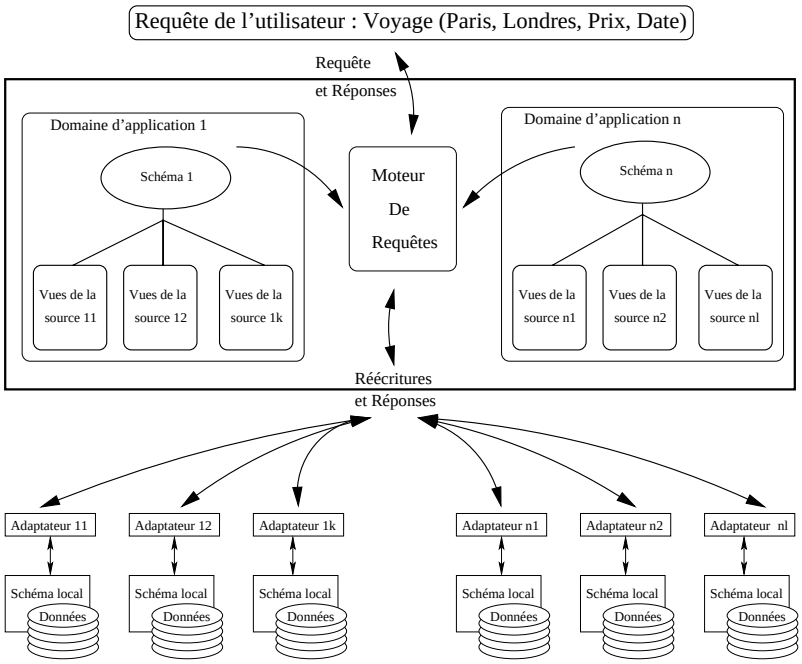

FIG. 1 – *Architecture d'un médiateur*

L'objectif de cet article est de présenter les choix de modélisation faits dans le cadre du projet PICSEL [28] et leurs conséquences sur le problème de la réécriture de requêtes en termes de vues. PICSEL se distingue des systèmes d'intégration d'informations existants par la possibilité d'exprimer le schéma global dans un langage (CARIN) combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (la logique de description $\mathcal{ALN}$).

De plus PICSEL est pour l'instant le seul système médiateur intégrant un module d'affinement de requêtes, première brique d'un module de dialogue coopératif entre un médiateur et ses utilisateurs. Ce module est déclenché quand l'ensemble des plans de requêtes calculé par l'algorithme de réécritures de requêtes en termes de vues est vide.

Le plan de l'article est le suivant : un panorama des systèmes existants d'intégration d'informations à base de médiateur est présenté dans la section 2. La section 3 est consacrée à la modélisation et à la représentation des connaissances dans PICSEL. Le calcul des plans de requêtes est détaillé et illustré dans la section 4. Le mécanisme d'affinement de requêtes est présenté dans la section 5. Finalement, nous concluons et dégageons quelques perspectives en section 6.

2 PANORAMA DES MÉDIATEURS EXISTANTS

Les différents systèmes d'intégration d'informations à base de médiateur se distinguent par :

1. d'une part, les langages utilisés pour modéliser le schéma global, les schémas des sources de données à intégrer et les requêtes des utilisateurs,

2. d'autre part, la façon dont est établie la correspondance entre le schéma global et les schémas des sources de données à intégrer.

Concernant le second point, on distingue deux manières d'établir la correspondance entre le schéma global et les schémas des sources de données à intégrer.

L'approche *Global As Views (GAV)* a été la première à être proposée pour l'intégration d'informations et provient du monde des bases de données fédérées. Elle consiste à définir le schéma global en fonction des schémas des sources de données à intégrer. Pour cela, les prédicats du schéma global, aussi appelés *relations globales*, sont définis comme des vues sur les prédicats des schémas des sources à intégrer. Cette approche suppose donc que les sources à intégrer soient connues à l'avance. Comme les requêtes d'un utilisateur s'expriment en termes des prédicats du schéma global, on obtient facilement une requête en termes des schémas des sources de données intégrées, en remplaçant les prédicats du schéma global par leur définition : on dit que l'on procède au *dépliement de la requête*. Cette opération de dépliement est effectuée par chaînage arrière lorsque les requêtes et les vues sont définies par des règles. Une fois dépliée, une requête peut alors être évaluée de façon standard sur les extensions des sources de données. Ainsi, la construction de la réponse à une requête dans une approche GAV se ramène à l'évaluation standard d'une requête, une fois sa reformulation par dépliement

effectuée. L'inconvénient de l'approche GAV est qu'elle est peu adaptée à l'ajout de nouvelles sources de données : cela peut nécessiter de mettre à jour la définition de l'ensemble des prédicats du schéma global.

L'approche *Local As Views (LAV)* est l'approche duale qui consiste à définir les schémas des sources de données à intégrer en fonction du schéma global. C'est l'approche suivie dans cet article. Les avantages et inconvénients de cette approche sont inversés par rapport à l'approche GAV. L'approche LAV est très flexible par rapport à l'ajout (ou la suppression) de sources de données à intégrer : cela n'a aucun effet sur le schéma global, seules des vues doivent être ajoutées (ou supprimées). Le prix à payer pour cette flexibilité et cette simplicité de mise à jour est la complexité de la construction des réponses à une requête dans un médiateur conçu selon l'approche LAV. La réécriture de requêtes en termes de vues est la seule possibilité pour obtenir des plans de requêtes exprimés en termes de vues (les réécritures), que l'on doit ensuite exécuter pour interroger les sources de données.

Nous structurons le panorama des principaux médiateurs en fonction du langage de représentation des connaissances utilisé pour exprimer le schéma global. Pour chacun des systèmes, nous indiquons s'il suit une approche GAV ou LAV.

2.1 Systèmes fondés sur un schéma global à base de règles

Les systèmes les plus représentatifs de cette famille sont : *Razor* ([18]), *Internet Softbot* ([16]), *Infomaster* ([19]) et *Information Manifold* ([22]). Ils suivent tous une approche LAV.

Les systèmes *Razor* et *Internet Softbot* sont les seuls qui utilisent le langage DATALOG pur (sans récursion) pour modéliser le schéma global et pour exprimer les requêtes de l'utilisateur. *Infomaster* et *Information Manifold* utilisent des extensions de DATALOG. *Infomaster* permet d'exprimer, en plus des règles DATALOG, des contraintes d'intégrité. Ces dernières peuvent être vues comme des règles concluant sur *Faux*, jouant un rôle particulier. Elles ne servent pas à inférer de nouveaux faits, mais à vérifier la cohérence des faits déduits. *Information Manifold* étend le cadre de DATALOG en permettant que certains prédicats utilisés dans les règles soient des concepts définis par ailleurs comme des classes, à l'aide de constructeurs de logiques de description. Le langage utilisé dans *Information Manifold* est un précurseur du langage CARIN considéré dans cet article.

Le système *HERMES* ([31]) est un système de bases de données fédérées et peut être vu comme un médiateur suivant une approche GAV. Il ne repose pas sur la description sémantique d'un domaine d'application ou du contenu de différentes bases de données. Son objectif est simplement de pouvoir combiner des requêtes posées à divers systèmes de gestion de bases de données. Il repose sur un langage de requêtes uniforme permettant d'englober et de combiner dans une même requête l'appel à des bases de données relationnelles, orientées-objet, spatiales, d'images, etc. Son langage de requêtes est un langage à base de règles qui est une sorte d'extension de Prolog par des annotations permettant de gérer l'incertitude et le temps.

2.2 Systèmes fondés sur un schéma global à base de classes

Le système *TSIMMIS* ([11]) est fondé sur un langage orienté-objet appelé OEM, pour le schéma global et les vues, ainsi que sur le langage OEM-QL pour les requêtes ([27]). Ce système suit une approche GAV.

Les systèmes *SIMS* ([4] et [3]) et *OBSERVER* ([24]) utilisent une logique de description pour décrire le schéma global, les vues et les requêtes. Ils relèvent d'une approche LAV puisqu'ils décrivent le contenu des sources à intégrer en fonction des descriptions de concepts du schéma global. *SIMS* utilise LOOM ([23]) alors qu'*OBSERVER* a fait le choix de CLASSIC ([9]). Dans ces deux systèmes, le problème de réécriture de la requête en termes des vues est traité comme un problème de planification et la complétude des algorithmes de réécriture n'est pas abordée.

Le système *MOMIS* ([5]) est fondé sur l'utilisation d'une logique de description très riche (ODL-I3) pour décrire les schémas des sources de données à intégrer. L'approche suivie est de type GAV puisque le schéma global est inféré à partir des schémas des sources.

2.3 Systèmes fondés sur un schéma global à base d'arbres

Avec l'avènement de XML, des médiateurs commencent à voir le jour au dessus de données semi-structurées ayant le format de documents XML. La médiation sémantique dans C-WEB ([13, 1]) est fondée sur THESAURI. Dans le médiateur de XYLEME ([30, 14]), le schéma global, support de l'interface de requêtes de haut niveau, est un ensemble de *DTDs abstraits* qui sont des

arbres de termes structurant le vocabulaire de domaines tels que la culture ou le tourisme. Les deux systèmes relèvent à la fois de l'approche GAV et LAV puisque la correspondance entre le vocabulaire du médiateur et le vocabulaire des sources est exprimée par de simples mappings de chemins.

PICSEL suit une approche LAV. Il se distingue des systèmes existants par la possibilité d'exprimer le schéma global dans un langage (CARIN) combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (la logique de description $\mathcal{ALN}$).

3 MODÉLISATION ET REPRÉSENTATION DES CONNAISSANCES DANS PICSEL

PICSEL permet à un utilisateur d'effectuer des recherches globales, c'est-à-dire dans les termes du schéma global du médiateur, ou ontologie. Un langage, appelé *langage de requêtes*, a été défini pour poser les requêtes. Par exemple, il permet à un utilisateur d'exprimer la recherche d'instances d'hôtels situés dans un lieu au soleil et équipés d'un golf. Ce langage manipule les termes de l'ontologie composée de l'ensemble des prédicats modélisant le domaine d'application du système. Sur l'exemple précédent, la requête est alors exprimée à l'aide des prédicats suivants : Hôtel, LieuAuSoleil, LrAvecGolf, SituéDans. Les réponses aux requêtes posées sont calculées par réécriture en termes de vues. Ces vues abstraites des données stockées dans les différentes sources d'information sont des requêtes prédéfinies, également exprimées en fonction de l'ontologie, dans un langage particulier, *le langage de vues*. Supposons, dans l'exemple, que deux sources soient accessibles, S1 décrivant des hôtels et leur localisation dans une île des Caraibes, S2 décrivant des équipements sportifs situés dans des hôtels. A chaque source est associée un ensemble de vues, c'est-à-dire de requêtes dont les extensions représentent l'ensemble des données qui peuvent être obtenues si on interroge la source. Ces requêtes sont exprimées par rapport à l'ontologie. Dans l'exemple, les sources S1 et S2 sont toutes deux exprimées, entre autres, par rapport au terme Hôtel. Le processus de réécriture permet d'identifier les sources pertinentes pour répondre à la requête, d'identifier les données qui y sont recherchées et la façon dont il faut les combiner pour répondre précisément. Ainsi, la réécriture de la requête de l'exemple combinera l'interrogation de la source S1 de façon à obtenir des hôtels situés dans une

île des Caraibes (lieu au soleil) avec l'interrogation de la source S2 pour ne retenir qu'un sous-ensemble d'hôtels, ceux permettant la pratique du golf.

Le formalisme CARIN-$\mathcal{ALN}$ a été adopté comme support de la description de l'ontologie du domaine ainsi que comme langage de requêtes. Le langage de vues n'utilise, lui, qu'une restriction de ce formalisme afin de garantir la décidabilité de la construction des plans de requêtes. CARIN-$\mathcal{ALN}$ est un formalisme de représentation de connaissances combinant dans un cadre logique homogène un langage de règles et un langage de classes. Le langage de classes considéré est la logique de descriptions $\mathcal{ALN}$ qui offre un certain nombre de constructeurs logiques pour décrire des classes, appelées des *concepts* pour des raisons historiques liées à l'origine des logiques de description (les réseaux sémantiques). L'intérêt de la logique de description $\mathcal{ALN}$ est que le test de subsomption est polynômial ([15]).

Dans la section 3.1, après avoir décrit précisément le formalisme CARIN-$\mathcal{ALN}$, nous présenterons le langages de vues et le langage de requêtes. Dans la section 3.2, nous montrerons comment la modélisation des connaissances utiles pour le médiateur a été guidée et contrainte par le formalisme.

3.1 Représentation des connaissances dans PICSEL

Nous commençons par décrire le langage CARIN-$\mathcal{ALN}$ choisi pour représenter l'ontologie du domaine d'application du médiateur, puis nous décrirons succinctement le langage de vues et le langage de requêtes permettant d'exprimer en termes des concepts de l'ontologie du domaine, respectivement, le contenu des sources et les requêtes des utilisateurs.

3.1.1 CARIN-$\mathcal{ALN}$: le langage de représentation de l'ontologie du domaine

Dans PICSEL, les connaissances du domaine sont exprimées de manière déclarative au travers de deux composantes : une composante terminologique et une composante déductive constituée d'un ensemble de règles.

La composante terminologique de CARIN-$\mathcal{ALN}$.

Elle comprend des définitions et des inclusions de concepts :

• Une définition de concept NC := *ConceptExpression* associe un nom de concept NC à une expression de concept de la logique de description $\mathcal{ALN}$.

Les concepts de base d'une terminologie sont ceux qui n'apparaissent pas en partie gauche d'une définition. Un nom de concept NC dépend d'un nom de concept NC' si NC' apparaît dans la définition de NC. Un ensemble de définitions de concept est dit *cyclique* s'il y a un cycle dans la relation de dépendance des noms de concept. Dans PICSEL, on impose que l'ensemble des définitions de concepts soit non cyclique.

• Une inclusion de concepts $C_1 \sqsubseteq C_2$ est une assertion exprimant le fait que le concept C_1 est inclus dans le concept C_2. 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. Cette deuxième forme d'inclusion permet d'exprimer, en utilisant le concept $\bot$, que les deux concepts A_1 et A_2 sont exclusifs.

La logique de description $\mathcal{ALN}$ contient les constructeurs de *conjonction* ($C_1 \sqcap C_2$), de *restriction de concept* ($\forall r\ C$) qui représente l'ensemble des éléments qui ne sont en relation par le rôle[2] r qu'avec des éléments du concept C, de *restrictions de cardinalité* ($\geq n\ r$), (respectivement ($\leq n\ r$)), qui représentent l'ensemble des éléments qui sont en relation par r avec au moins, (respectivement au plus), n éléments distincts, et de *négation* $\neg$ (sur les concepts de base uniquement).

La sémantique logique d'$\mathcal{ALN}$ associe une relation unaire à chaque concept et une relation binaire à chaque rôle de façon à satisfaire les équivalences suivantes, où x est une variable libre représentant une instance quelconque de concept ($C(x)$ est la traduction logique de l'appartenance de x au concept C) :

$(C_1 \sqcap C_2)(x) \equiv C_1(x) \wedge C_2(x)$

$((\forall r\ C))(x) \equiv \forall$ y (r(x,y) $\Rightarrow$ C(y))

$((\geq n\ r))(x) \equiv \exists y_1 \ldots \exists y_n$ (r(x, y_1) $\wedge \ldots \wedge$ r(x, y_n) $\wedge \bigwedge_{\{i,j|i\neq j\}} y_i \neq y_j$)

$((\leq n\ r))(x) \equiv \forall y_1 \ldots \forall y_{n+1}$ (r(x, y_1) $\wedge \ldots \wedge$ r(x, y_{n+1}) $\Rightarrow \bigvee_{\{i,j|i\neq j\}} y_i = y_j$)

(¬A)(x) $\equiv \neg A$(x).

La sémantique logique d'une terminologie $\mathcal{T}$ en découle de la façon suivante.

2. En logique de description, la notion de rôle a une signification bien particulière. Elle correspond à une relation binaire.

- Toute définition de concept $NC := ConceptExpression$ est interprétée logiquement comme : $\forall x(NC(x) \Leftrightarrow ConceptExpression(x))$.
- Toute inclusion de concept $A \sqsubseteq ConceptExpression$ est interprétée logiquement comme : $\forall x(A(x) \Rightarrow ConceptExpression(x))$.
- Toute inclusion de concept de la forme $A_1 \sqcap A_2 \sqsubseteq \bot$ est interprétée logiquement comme : $\neg\,(\exists\, x(A_1(x) \wedge A_2(x)))$.

Les deux définitions suivantes de la *subsomption* entre concepts et de la *satisfiabilité* de concepts sont également fondées sur la sémantique logique.

Définition 1 (subsomption entre concepts)
Soit $\mathcal{T}$ une composante terminologique, et C_1 et C_2 deux concepts définis dans $\mathcal{T}$: C_1 est subsumé par C_2 modulo $\mathcal{T}$ *(noté $C_1 \preceq_{\mathcal{T}} C_2$) si et seulement si : $\mathcal{T} \models \forall x(C_1(x) \Rightarrow C_2(x))$.*

Définition 2 (satisfiabilité de concepts)
Soit $\mathcal{T}$ une composante terminologique, et C un concept défini dans $\mathcal{T}$: C est satisfiable modulo $\mathcal{T}$ *si et seulement si : $\mathcal{T} \models \exists x C(x)$.*

Dans le cadre de PICSEL, le module de classification ONTOCLASS, codé en Java, effectue de façon très efficace le test de satisfiabilité pour chaque concept de la composante terminologique et le test de subsomption sur chaque couple de concepts.

Exemple 1
Soit $\mathcal{T}$ la terminologie composée des quatre définitions de concepts et des deux inclusions suivantes :

Compagnie1 := ($\geq$ 1 associé) $\sqcap$ ($\forall$associé CompagnieAméricaine).
Compagnie2 := ($\leq$ 2 associé) $\sqcap$ ($\forall$associé CompagnieEuropéenne).
Compagnie3 := ($\leq$ 1 associé) $\sqcap$ ($\forall$associé CompagnieFrançaise).
Compagnie4 := Compagnie1 $\sqcap$ Compagnie2.
CompagnieFrançaise $\sqsubseteq$ CompagnieEuropéenne.
CompagnieAméricaine $\sqcap$ CompagnieEuropéenne $\sqsubseteq \bot$.

Le logiciel ONTOCLASS détectera que le concept Compagnie4 est insatisfiable, mais aussi que Compagnie3 est subsumé par Compagnie2. $\square$

La composante déductive de CARIN-$\mathcal{ALN}$**.**

Elle comprend :

• Un ensemble $\mathcal{D}_r$ de règles saines[3] de la forme : $p_1(\bar{X}_1)\wedge\ldots\wedge p_n(\bar{X}_n) \rightarrow q(\bar{X})$, où $\bar{X}$,$\bar{X}_1 \ldots \bar{X}_n$ sont des vecteurs de variables tels que l'ensemble $var(\bar{X})$ des variables apparaissant dans le vecteur $\bar{X}$ de la conclusion de la règle soit inclus dans l'union $var(\bar{X}_1)\cup\ldots\cup var(\bar{X}_n)$ des variables apparaissant en condition de la règle. Les règles permettent de définir des prédicats q (d'arité quelconque) en fonction d'autres prédicats $p_1, \ldots, p_n$. On distingue les *prédicats de base* qui sont les prédicats qui n'apparaissent pas en conclusion de règles. Parmi les prédicats de base, certains (unaires ou binaires) sont des noms de concepts ou de rôles intervenant dans la composante terminologique.

• Un ensemble $\mathcal{C}_r$ de contraintes d'intégrité ayant la forme de règles[4] concluant sur $\bot$:

$p_1(\bar{X}_1) \wedge \ldots \wedge p_m(\bar{X}_m) \rightarrow \bot$, où les p_i sont des prédicats d'arité quelconque.

La sémantique logique associée à la composante déductive consiste à interpréter :

- chaque règle $p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \rightarrow q(\bar{X})$ par l'implication logique : $\forall X_1, \ldots, \forall X_k(\exists Y_1, \ldots, \exists Y_l\, p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{X}))$, où $X_1, \ldots, X_k$ sont les variables de $\bar{X}$, et $Y_1, \ldots, Y_l$ sont les variables apparaissant en condition de la règle sans faire partie des variables $\bar{X}$ de la conclusion.

- et chaque contrainte d'intégrité $p_1(\bar{X}_1) \wedge \ldots \wedge p_m(\bar{X}_m) \rightarrow \bot$ par la formule logique :
$\neg(\exists Y_1, \ldots \exists Y_l\, (p_1(\bar{X}_1) \wedge \ldots \wedge p_m(\bar{X}_m)))$, où $Y_1, \ldots, Y_l$ sont les variables de la contrainte d'intégrité.

Dans la suite, on aura besoin de distinguer trois sortes d'atomes pouvant apparaître en partie condition d'une règle (ou dans la définition d'une requête comme on le verra plus loin) : les *atome-concepts*, les *atome-rôles* et les *atomes ordinaires*.

– Un *atome-concept* est un atome de la forme $C(U)$ où U est une va-

3. Une règle est saine si toutes les variables apparaissant dans sa conclusion apparaissent aussi dans sa partie condition

4. Des inégalités entre variables peuvent être spécifiées dans une contrainte d'intégrité. Nous ne les faisons pas apparaitre pour simplifier la présentation et ne pas alourdir les notations.

riable et C est un nom de concept défini dans la composante terminologique.

– Un *atome-rôle* est un atome de la forme $r(U_1, U_2)$ où U_1 et U_2 sont des variables et r est un nom de rôle apparaissant dans des définitions de concepts de la composante terminologique.

– Un *atome ordinaire* est un atome de la forme $p(U_1, \ldots, U_n)$ où $(U_1, \ldots, U_n)$ est un vecteur de variables et p un prédicat d'arité n quelconque n'apparaissant pas dans la composante terminologique.

3.1.2 Le langage de vues

Le contenu de chaque source $\mathcal{S}$ est représenté à partir du vocabulaire $\mathcal{V}_S$ constitué d'autant de noms de vues v_i qu'il y a de relations du domaine dont on sait que la source $\mathcal{S}$ fournit des instances. La description du contenu d'une source $\mathcal{S}$ en terme de vues est constituée de :

1) un ensemble $\mathcal{I}_s$ de correspondances $v_i \rightarrow p$ reliant chaque vue à la relation p du domaine dont elle peut fournir des instances,

2) un ensemble $\mathcal{C}_s$ de contraintes sur les vues, de la forme :

• $v \sqsubseteq C$ où C est une expression de concept, avec, pour chaque vue v, au plus une contrainte de cette forme,

• $l_1(\bar{X}_1) \wedge \cdots \wedge l_n(\bar{X}_n) \rightarrow \bot$, où chaque $l_i(\bar{X}_i)$ est soit un atome-vue $v_i(\bar{X}_i)$, soit la négation d'un atome-vue, de sorte que chaque contrainte de cette forme contienne au plus une négation d'atome-vue.

Exemple 2
Soit la source S_1 décrivant des hôtels et leur localisation dans une île des Caraibes. Les implications suivantes expriment le fait que la source S_1 permet d'obtenir des instances du concept Hotel ainsi que des instances du rôle SituéDans.
$\mathcal{I}_{S1}$:
S_1-hotel $\rightarrow$ Hotel
S_1-situation $\rightarrow$ SituéDans

Les contraintes suivantes traduisent le fait que les hôtels trouvées en accédant à la source S_1 sont tous situés dans une île des Caraibes et que les instances de S_1-situation concernent les hôtels de la source S_1.

$\mathcal{C}_{S1}$:
S_1-hotel $\sqsubseteq$ ($\forall$SituéDans Caraibes)
S_1-situation(x,y) $\wedge \neg$ S_1-hotel(x) $\rightarrow \bot$ □

3.1.3 Langage de requêtes

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 autorisées sont des *requêtes conjonctives* de la forme :

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

où les p_i sont des atome-concepts, des atome-rôles ou des atomes ordinaires. Leur conjonction constitue le *corps de la requête*, alors que $Q(\bar{X})$ est appelé la *tête de la requête*.
Les variables du vecteur $\bar{X}$ 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.

Exemple 3
La requête suivante exprime la recherche d'hôtels situés dans un lieu au soleil et équipés d'un golf.
Q(x): Hôtel(x) $\wedge$ LrAvecGolf(x) $\wedge$ SituéDans(x, l) $\wedge$ LieuAuSoleil(l)
. □

Une requête conjonctive peut être vue comme une règle de CARIN-$\mathcal{ALN}$ dont la partie condition est le corps de la requête et dont la conclusion est la tête de la requête.

Nos algorithmes travaillent sur des requêtes dont le corps est *normalisé*. La *normalisation* d'une conjonction d'atomes consiste à remplacer la conjonction $C_1(u) \wedge \cdots C_k(u)$ de tous les atomes-concepts portant sur la même variable u par un seul atome-concept $(C_1 \sqcap \cdots \sqcap C_k)(u)$.

Nous décrivons, dans la section qui suit, les choix de modélisation pour représenter l'ontologie du domaine dans le langage logique CARIN-$\mathcal{ALN}$.

3.2 De la modélisation à la représentation du domaine guidée par le langage

La démarche adoptée pour construire l'ontologie a consisté à concevoir dans un premier temps le modèle sous forme papier.

Les connaissances de l'ontologie sont principalement vues comme le support de l'interface entre le système et ses utilisateurs qui sont, soit les usagers qui posent des requêtes, soit les administrateurs qui décrivent le contenu des sources d'information. Tous les termes susceptibles d'être employés par ces deux types d'utilisateurs doivent donc être représentés. Dans la phase de modélisation, nous avons estimé les besoins des usagers en simulant nous-mêmes leur rôle. Les besoins pour décrire les sources ont été évalués à partir des informations présentes dans les catalogues de tourisme, disponibles sous forme papier ou électronique, dans les agences de voyage ou sur le Web. Nous avons tenté de modéliser une très grande partie des informations exprimées dans ces catalogues. En effet, les concepteurs des documents les ont jugées nécessaires pour décrire leurs produits, vraisemblablement parce qu'elles correspondent aux critères de choix des usagers. Dans les catalogues exploités, beaucoup d'informations ne sont pas réellement structurées et figurent sous forme de commentaires. Notre travail a consisté à les analyser pour que les concepts sous-jacents apparaissent explicitement dans le modèle. Par exemple, il nous a semblé utile de permettre à un administrateur d'une source d'information d'entrer explicitement le nombre de pistes de ski alpin d'une station classées noires, ou les caractéristiques des différents forfaits, et également de permettre à l'usager de vérifier aisément qu'il s'agit d'une station familiale dans laquelle ses jeunes enfants pourront bénéficier d'activités adaptées.

Le modèle obtenu a été représenté à l'aide d'une notation structurée qui explicite les noms de concepts et de relations retenus, décrit des hiérarchies de concepts à représenter et énonce des définitions de concepts. Sa représentation en CARIN-$\mathcal{ALN}$ a été effectuée en utilisant le cadre de représentation du langage qui, en contraignant la représentation, a permis de la guider. Nous décrivons dans les sections qui suivent comment le modèle de l'ontologie a pu être représenté en CARIN-$\mathcal{ALN}$ et comment ce cadre de travail a conduit à affiner et à enrichir ce modèle.

3.2.1 Ce qu'on exprime dans la partie terminologique

a) La définition des concepts et des rôles

Le modèle de l'ontologie comprend une hiérarchie principale de concepts dont la racine est le concept produit , qui représente ce qui peut se vendre dans le domaine du tourisme et qui regroupe les logements, trajets, locations de véhicule, stages. Le modèle comprend, par ailleurs, des hiérarchies secondaires disjointes décrivant des catégorisations d'objets de sous-domaines du domaine d'application (lieu, loisir, prestation, service, équipement). Tous ces concepts sont représentés dans une terminologie $\mathcal{ALN}$.
Chaque concept est défini au travers de ses relations avec d'autres concepts. Pour un concept donné, le modèle précise le concept qui le généralise (concept père dans la hiérarchie) et éventuellement ses propriétés spécifiques ou bien l'ensemble des propriétés nécessaires et suffisantes d'une entité pour appartenir à ce concept. Les propriétés sont représentées à l'aide de rôles et de constructeurs de restriction de concept ($\forall$) et de cardinalité ($\geq$, $\leq$). Les exemples ci-dessous illustrent la représentation en CARIN-$\mathcal{ALN}$ des concepts. L'exemple 4.a représente la définition du concept produit : un produit a un seul prix, une seule date de début, éventuellement des services ou prestations associés. L'exemple 4.b représente la définition du concept activitéSportive : une activitéSportive est une activité dont la nature associée peut être qualifiée de loisirSportif. Produit et activitéSportive sont tous deux des concepts dits définis en logique terminologique. L'exemple 4.c illustre l'inclusion du concept de base equipementCulturel dans le concept equipement.

Exemple 4

a. (Produit :=
(≥ 1 prixAssocié) ⊓ (≤ 1 prixAssocié)
⊓ (∀prixAssocié Nombre)
⊓ (≥ 1 dateDébutAssocié) ⊓ (≤ 1 dateDébutAssocié)
⊓ (∀dateDébutAssocié Date)
⊓ (∀serviceProduitAssocié Service)
⊓ (∀prestationProduitAssocié Prestation))

b. (ActivitéSportive :=
(Activité ⊓ (∀natureActivitéAssocié LoisirSportif)))

c. (EquipementCulturel ⊑ Equipement) □

Selon ce cadre de représentation, un concept "isolé", sans relation avec un autre concept, est dépourvu de sens. Il est éliminé de la représentation.

Les propriétés, traduites par des rôles, issues du modèle de l'ontologie et représentées dans cette étape, sont de deux types :

• les propriétés qu'un utilisateur du serveur souhaiterait préciser lors de l'expression d'une requête ou lors de la description du contenu d'une source. Exemple de requête : une chambre d'hôtel avec 2 lits une place et située dans un hôtel avec piscine,

• les propriétés nécessaires pour structurer et articuler les concepts entre eux. Ainsi, la propriété lieuDeRésidenceAssocié permet de relier un logement (ex. : une chambre d'hôtel) à un lieuDeRésidence (ex. : un 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, équipements) où le logement est situé.

Ce cadre de travail a facilité la représentation des concepts et des rôles de l'ontologie. D'une part, le mécanisme d'inclusion de concepts permet à un concept d'apparaître comme une spécialisation d'un autre concept et d'hériter de ses propriétés sans qu'il soit nécessaire de le définir plus précisément. Ainsi, si le concept sportDeMontagne a été défini comme un sport praticable dans un lieuAvecMontagne, on peut en définir des spécialisations par inclusion de concepts (skiAlpin, luge) qui hériteront de ses propriétés. Par ailleurs, le langage rend possible la représentation de points de vue multiples sur un concept qui, de ce fait, hérite des propriétés de l'ensemble des concepts plus généraux. Ainsi, le concept bateau peut être défini comme étant à la fois un moyenDeTransport et un lieuDeRésidence. Enfin, toute une hiérarchie de concepts peut être définie sans qu'aucune propriété ne soit explicitée pour aucun d'entre eux. Ainsi, la hiérarchie des équipements décrit des catégories d'équipements de toutes sortes dont les équipements sportifs, parmi lesquels on trouve, entre autres, les piscines qu'on peut spécialiser en piscineAVague et piscineEauDeMer sans qu'il soit nécessaire de définir ces concepts plus précisément.

Le concepteur est assisté durant tout le processus de représentation de l'ontologie par ONTOCLASS qui classifie automatiquement les concepts à partir de leur définition. La hiérarchie résultant de cette classification peut être visualisée. Cela aide à détecter d'éventuelles anomalies, guide pour effectuer des modifications et peut faire émerger des concepts généraux par la mise en évidence de propriétés communes à plusieurs concepts. Par exemple, la création du concept logement a été effectuée en factorisant les propriétés communes des concepts chambre et appartement. Ces deux concepts héritent des propriétés associées au concept général logement et seules leurs

propriétés spécifiques leur restent attachées.

Le cadre de travail a également conduit à renommer certains éléments, comme les noms de rôles. En effet, en logique terminologique, un rôle n'est pas défini en fonction des concepts qu'il lie. Il est alors tout à fait possible d'utiliser le nom serviceAssocié en tant que rôle de produit, de lieuDeRésidence ou de station. De ce fait, nous avons fait le choix d'utiliser des noms de rôles identiques lorsque les concepts concernés étaient liés par un lien de généralisation-spécialisation et nous avons préféré utiliser des noms de rôles distincts dans le cas contraire, afin d'éviter d'éventuelles ambiguïtés pour l'utilisateur du système.

b) L'expression de contraintes

Deux formes de contraintes ont été représentées dans la composante terminologique de CARIN-$\mathcal{ALN}$:

1) l'expression de relations d'exclusion entre concepts de base. L'exemple 5 ci-dessous traduit le fait qu'un équipement culturel ne peut pas être un équipement sportif.

Exemple 5

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

2) l'expression de contraintes de typage sur les rôles par le biais du constructeur de restriction de concepts. Dans l'exemple 4.a définissant le concept produit, la restriction ($\forall$dateDébutAssocié date) signifie que le rôle dateDébutAssocié ne relie des éléments de type produit qu'à des éléments de type date.

3.2.2 Ce qu'on exprime à l'aide de la partie déductive

a) Le recours aux règles

L'emploi de règles déductives permet de définir des propriétés sur les concepts par des prédicats déductibles n'appartenant pas à la composante terminologique. Suivant les cas, ce recours aux règles est une nécessité (cf. cas 1 à 4) ou une facilité d'écriture (cf. cas 5). Rappelons que le choix de ne pas offrir le constructeur de disjonction ou la possibilité d'exprimer des relations inverses dans la logique de description choisie, est justifié par la volonté de garantir la polynomialité du test de subsomption.

• Cas 1. Pour exprimer des relations autres que des relations unaires et binaires.
La règle R_1 définit la relation de nom VolAR et d'arité 4 à partir d'un certain nombre de relations de base. Elle exprime qu'un vol aller-retour (VolAR) est composé de deux vols consécutifs dans le temps. Le premier vol est au départ d'une ville villeDépart à la date dateDépart1 et à destination d'une ville villeArrivée. Le second vol est au départ de la ville villeArrivée à la date dateDépart2 et à destination de la ville villeDépart. De telles relations naires sont utiles pour formuler des requêtes exprimant que l'utilisateur recherche un vol aller-retour et désire connaître la ville de départ (villeDépart), la date de départ (dateDépart1), la ville d'arrivée (villeArrivée), la date de retour (dateDépart2) pour chaque vol obtenu en réponse.

Exemple 6
R_1 : Vol(v1) $\wedge$ lieuDépart(v1, villeDépart)
$\wedge$ lieuArrivée(v1, villeArrivée) $\wedge$ dateDépart(v1, dateDépart1)
$\wedge$ Vol(v2) $\wedge$ lieuDépart(v2, villeArrivée)
$\wedge$ lieuArrivée(v2, villeDépart) $\wedge$ dateDépart(v2, dateDépart2)
$\wedge$ dateAntérieure(dateDépart1, dateDépart2)
$\rightarrow$ VolAR(villeDépart, dateDépart1, villeArrivée, dateDépart2). □

• Cas 2. Pour construire une définition de relation disjonctive.
Les règles R_2 et R_3 traduisent qu'un produit pour jeune (produitJeune) est un produit auquel on associe soit au plus un service, soit des services bon marché. Cette relation ne peut être définie dans la partie terminologique du langage du fait de l'absence du constructeur de disjonction $\vee$.

Exemple 7
R_2 : produit (x) $\wedge$ ($\leq$ 1 produitServiceAssocié) $\rightarrow$ ProduitJeune(x)
R_3 : produit (x) $\wedge$ produitServiceAssocié(x,y) $\wedge$ bonMarché(y)
$\rightarrow$ ProduitJeune(x) □

• Cas 3. Pour exprimer des contraintes "non exclusives".
Le constructeur de restriction de concepts ($\forall$R C) permet d'exprimer des contraintes sur le domaine des concepts en relation avec d'autres, mais ces contraintes sont "exclusives". Ainsi, dans la définition suivante "lieuAvec-Montagne := lieu $\sqcap$ ($\forall$loisirPraticable sportDeMontagne)", la restriction se

lit “tous les loisirs praticables dans un lieuAvecMontagne sont des sportDeMontagne ”, ce qui exclut la natation ou le tennis. Si on souhaite représenter qu'il est possible de faire du ski et de la natation dans un lieuAvecMontagne équipé d'une piscine, on ne doit pas utiliser de restriction de concept, mais des règles. Les loisirs praticables dans un lieu peuvent y être définis comme dépendant à la fois des caractéristiques physiques du lieu considéré (cf. R_4 et R_5) et de ses équipements sportifs (cf. R_6).

Exemple 8
R_4 : lieuAvecMontagne(l) $\wedge$ sportDeMontagne(s)
$\rightarrow$ loisirPraticable(l, s).
R_5 : lieuAvecPlage(l) $\wedge$ sportNautique(s) $\rightarrow$ loisirPraticable(l, s).
R_6 : lieuAvecPiscine(l) $\wedge$ natation(s) $\rightarrow$ loisirPraticable(l, s). □

• Cas 4. Pour exprimer une relation inverse.

Dans PICSEL les définitions de concepts sont non cycliques et les rôles sont orientés. Par exemple, le rôle situationAssocié relie le concept lieuDeRésidence au lieu dans lequel il est situé. On s'interdit d'introduire, dans la partie terminologique, le rôle inverse qui donnerait les lieuDeRésidence se trouvant dans un lieu. En revanche, cette définition est possible par une règle (cf. R_7).

Exemple 9
R_7 : lieuDeResidence(r) $\wedge$ situationAssocié(r, l) $\wedge$ lieu(l)
$\rightarrow$ LrSeTrouvantDansLieu(l,r). □

• Cas 5. Pour traduire un raccourci d'enchaînement de rôles.

Une règle peut associer directement à un concept une propriété qui lui est déjà associée indirectement, par enchaînement de plusieurs rôles. Le but est d'éviter les redondances dans la composante terminologique. Ainsi, la règle R_8 permet de dériver le prédicat StationOuSeTrouveLogement qui associe directement à Logement, à partir des propriétés présentes dans la composante terminologique, le nom de la station de ski où se trouve son lieu de résidence associé.

Exemple 10
R_8 : logement(l) $\wedge$ lieuDeResidenceAssocié(l,r)
$\wedge$ stationLieuResidenceAssocié(r, st) $\wedge$ stationSki(st)
$\wedge$ nomAssocié(st,n) $\rightarrow$ StationOuSeTrouveLogement(l,n). □

b) L'utilisation de contraintes

Comme on l'a vu en 3.2.1, l'expression de contraintes d'exclusion entre concepts de base se fait en CARIN-$\mathcal{ALN}$ dans la composante terminologique. Les contraintes portant sur d'autres types de prédicats doivent être représentées par des contraintes d'intégrité.

• Contraintes traduisant des dépendances fonctionnelles.

La contrainte ci-dessous exprime le fait qu'un numéro de téléphone est propre à un seul lieu de résidence.

Exemple 11

NumTél (x_1,y) $\wedge$ lieuDeRésidence(x_1)
$\wedge$ NumTél (x_2,y) $\wedge$ lieuDeRésidence(x_2) $\wedge x_1 \neq x_2 \rightarrow \bot$ □

• Contraintes de typage sur les arguments de prédicats.

Les contraintes de typage s'écrivent $P(\bar{X}) \wedge \neg C_i(x_i) \rightarrow \bot$ avec $\bar{X} = (x_1, x_2, ..., x_i, ...x_n)$. Elles permettent de préciser le domaine de valeurs (ensemble des éléments de C_i) de chaque argument x_i de la relation P. Ainsi, selon l'exemple 12, la relation serviceAssocie établit toujours un lien avec un service.

Exemple 12

ServiceAssocié (x,y) $\wedge \neg$ Service(y) $\rightarrow \bot$ □

Les contraintes de typage peuvent remplacer une restriction de concept dans une définition. Ainsi, lorsqu'un rôle est utilisé dans la définition de plusieurs concepts, avec la même restriction de concept (ex. : ($\forall$serviceAssocié Service)), l'expression d'une contrainte (cf. exemple 12) évite de répéter la restriction de concept dans chaque définition.

Etant donné un certain rôle, contrainte et restriction de concept associé à ce rôle ne sont cependant pas toujours équivalentes. Lorsque le concept dans la contrainte est un concept généralisant celui spécifié dans la restriction de concept, contrainte et restriction de concepts peuvent coexister. La restriction de concept est utile car elle est plus précise que la contrainte. La contrainte est aussi utile. Elle s'applique en l'absence de restriction plus précise (cf. exemple 13).

Exemple 13
ChambreD'Hôtel := (Chambre ⊓ (∀lieuDeRésidenceAssocié Hôtel))
CabineBateau := (Chambre ⊓ (∀lieuDeRésidenceAssocié Bateau))
Le premier concept qui généralise à la fois Hôtel et Bateau est LieuDeRésidence. On peut donc écrire la contrainte qui suit :
lieuDeRésidenceAssocié(x,y) ∧ ¬ lieuDeRésidence(y) → ⊥.
Cette contrainte s'applique chaque fois que le rôle lieuDeRésidenceAssocié est utilisé dans une définition de concept sans qu'aucune restriction de concept ne soit indiquée. □

• Contraintes d'exclusion entre prédicats quelconques.

Ces contraintes sont de la forme $p_1(\bar{X}_1) \wedge ... \wedge p_n(\bar{X}_n) \rightarrow \perp$. Elles sont utiles pour exprimer des contraintes sémantiques sur le domaine d'application. Ainsi, l'exemple 14 a. exprime qu'il n'y a pas de vol direct entre deux pays en conflit. Elles servent également au traitement des exceptions. Dans l'exemple 14 b., CabineBateau est une spécialisation du concept Chambre et hérite à ce titre de toutes ses propriétés associées, donc, entre autres, des prestations. La contrainte de l'exemple 14 b. traduit le fait que Terrasse fait exception à cette règle car les terrasses ne font pas partie des prestations associées à une cabine de bateau.

Exemple 14
a. Pays(P_1) ∧ Pays(P_2) ∧ EnConflit(P_1,P_2) ∧ TrajetReliant(T, P_1,P_2) ∧ VolDirect(T) → ⊥
Pays et VolDirect sont des concepts, EnConflit est un rôle, TrajetReliant est un prédicat.

b. CabineBateau(L) ∧ prestationChambreAssocié(L, P) ∧ Terrasse(P) → ⊥
CabineBateau et Terrasse sont des concepts, prestationChambreAssocié est un rôle. □

3.2.3 Les choix de représentation dus aux limites du langage

Le langage CARIN-$\mathcal{ALN}$ a été choisi pour garantir l'obtention d'un système efficace. Ce langage s'avère, en règle générale, relativement riche, notamment parce qu'il s'agit d'un langage hybride combinant à la fois logique de description et règles Datalog non récursives. Lorsqu'une connaissance ne

peut s'exprimer à l'aide de la composante terminologique, il est souvent possible, comme le montrent les sections qui précèdent, de la représenter sous forme de règle ou de contrainte. Néanmoins, d'autres solutions ont parfois dû être imaginées pour pallier certaines limites du langage, par exemple l'absence du quantificateur existentiel $\exists$ et de la négation complète (le constructeur $\neg$ est restreint aux seuls concepts de base).

CARIN-$\mathcal{ALN}$ ne permet pas l'utilisation du quantificateur existentiel.

Ce manque a pu être comblé par l'utilisation de multiples rôles spécialisés. Ainsi pour définir une stationFamiliale comme une station équipée à la fois d'une garderie enfants et d'un centre médical, deux rôles distincts : garderieSurPlaceAssocie et médecinSurPlaceAssocie (cf. exemple 15) ont été introduits alors que le seul rôle serviceAssocie aurait pu suffire s'il avait été possible de l'utiliser dans des expressions comportant le quantificateur $\exists$ du type : ($\exists$ serviceAssocié GarderieEnfants) et ($\exists$ serviceAssocié Médecin) même si GarderieEnfants et Médecin sont définis comme des concepts disjoints.

Exemple 15

stationFamiliale := station
- $\sqcap$ ($\geq$ 1 garderieSurPlaceAssocié)
- $\sqcap$ ($\forall$garderieSurPlaceAssocie GarderieEnfants)
- $\sqcap$ ($\geq$ 1 médecinSurPlaceAssocié)
- $\sqcap$ ($\forall$médecinSurPlaceAssocie Médecin) □

La négation autorisée dans CARIN-$\mathcal{ALN}$ est très restreinte.

En particulier, elle ne peut pas s'appliquer aux concepts définis. Nous avons contourné cette limitation en multipliant les noms de concepts et en exploitant les contraintes d'intégrité. Ainsi, dans l'exemple 16, le concept lieuSansMontagne a été introduit de façon à exclure la pratique d'un SportDeMontagne ailleurs que dans un lieuAvecMontagne. lieuAvecMontagne et lieuSansMontagne sont par ailleurs déclarés disjoints par une contrainte d'intégrité (cf. exemple 16 b).

Exemple 16

a. lieuSansMontagne(l) $\wedge$ sportDeMontagne(s)
   $\wedge$ loisirPraticable(l, s) $\rightarrow \bot$

b. lieuAvecMontagne(x) $\wedge$ lieuSansMontagne(x) $\rightarrow \bot$ $\square$

A l'issue de cette étape, nous disposons d'une première version opérationnelle de l'ontologie composée d'environ 200 concepts et 300 rôles. Celle-ci respecte les contraintes du langage et le modèle de l'ontologie préalablement construit. Son contenu a été défini en phase de modélisation et complété dans cette première étape de représentation, compte tenu des fonctionnalités du médiateur à mettre en place.

4 CALCUL DES PLANS DE REQUÊTES DANS PICSEL

Nous commençons par définir formellement ce qu'est un plan de requêtes (requête conjonctive définie par une conjonction d'atomes-vues) pour une requête exprimée par un utilisateur en fonction du vocabulaire de l'ontologie du domaine. Nous donnons ensuite les principales étapes du calcul des plans de requêtes. Nous détaillerons plus particulièrement l'étape de réécriture d'un atome-concept qui constitue la principale contribution algorithmique de ce travail et a permis d'obtenir le premier algorithme de réécriture de requêtes conjonctives contenant des expressions de concepts et de rôles ([21, 20]).

4.1 Définition d'un plan de requêtes

Dans cette section, pour les vues v en correspondance avec des concepts p, on regroupe dans la définition de v la correspondance avec p et les contraintes terminologiques qui lui sont éventuellement associées.

Définition 3 (Définition d'une vue)
Soit v une vue de $\mathcal{V}$ faisant partie de la description $\mathcal{I}_\mathcal{S} \cup \mathcal{C}_\mathcal{S}$ d'une source $\mathcal{S}$, et telle que $v \rightarrow p \in \mathcal{I}_\mathcal{S}$. La définition de la vue v, *notée $def(v)$, est :*

- $def(v) = p \sqcap C$ *si* $v \sqsubseteq C \in \mathcal{C}_\mathcal{S}$,
- $def(v) = p$ *si v n'est associée à aucune contrainte terminologique.*

Définition 4 (Expansion de requêtes conjonctives sur des atomes-vues)
- *Une* requête conjonctive sur des atomes-vues *est une requête $q_\mathcal{V}$ définie par une règle dont le corps est une conjonction d'atomes-vues*[5] *:* $q_\mathcal{V}(\bar{X}) : v_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge v_n(\bar{X}_n, \bar{Y}_n)$.
- L'expansion *d'une requête $q_\mathcal{V}$, notée $expansion(q_\mathcal{V})$, est la conjonction d'atomes obtenue à partir du corps de $q_\mathcal{V}$ en remplaçant chaque vue v par sa définition $def(v)$.*

Un *plan de requêtes* pour une requête q (aussi appelé *réécriture* de q en termes de vues) est une requête conjonctive sur des atomes-vues dont l'expansion est d'une part satisfiable, et d'autre part implique (modulo la terminologie et la base de règles) le corps de la requête q.

Nous commençons par définir la satisfiabilité (modulo l'ontologie du domaine) d'une conjonction d'atomes normalisée. Nous définissons ensuite formellement la notion de plan de requêtes.

Définition 5 (Satisfiabilité d'une conjonction d'atomes)
Soit cj une conjonction d'atomes normalisée. Soit une ontologie composée d'une composante terminologique $\mathcal{T}$ et d'une composante déductive $\mathcal{R} = D_r \cup C_r$. cj est satisfiable (modulo $\mathcal{T}$ et $\mathcal{R}$) *ssi:*
- *pour tout atome-concept $C(u)$ de cj, C est satisfiable modulo $\mathcal{T}$, et*
- *$\mathcal{R}, cj \not\models \bot$.*

Définition 6 (Plan de requêtes (aussi appelé réécriture))
Soit une ontologie définissant le vocabulaire d'un domaine et composée d'une composante terminologique $\mathcal{T}$ et d'une composante déductive $\mathcal{R} = D_r \cup C_r$. Soit $\mathcal{V}$ un ensemble de vues définies sur ce vocabulaire. Soit q une requête conjonctive également définie sur ce même vocabulaire : $q(\bar{X}) : p_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge p_n(\bar{X}_n, \bar{Y}_n)$.
Une requête conjonctive $q_\mathcal{V}$ est un plan de requêtes *pour q (aussi appelé* réécriture *de q) ssi :*
- *$expansion(q_\mathcal{V})$ est satisfiable modulo $\mathcal{T}$,*
et
- *$\mathcal{R}, \mathcal{T}, expansion(q_\mathcal{V}) \models \exists \bar{Y}_1 \ldots \exists \bar{Y}_n [p_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge p_n(\bar{X}_n, \bar{Y}_n)]$.*

5. Le corps d'une requête conjonctive sur des atomes-vues peut en fait contenir des inégalités entre variables que nous ne faisons pas apparaitre dans cette définition pour en simplifier la présentation et ne pas alourdir les notations.

Exemple 17
Soit le sous-ensemble suivant de la terminologie :

$\mathsf{hôtelDeLuxe} \sqsubseteq \mathsf{hôtel}$	$\mathsf{france} \sqsubseteq \mathsf{lieu}$
$\mathsf{lieuAvecMontagne} \sqsubseteq \mathsf{lieu}$	$\mathsf{alpesFrancaises} \sqsubseteq \mathsf{france}$
$\mathsf{régionPACA} \sqsubseteq \mathsf{france}$	$\mathsf{alpesItaliennes} \sqsubseteq \mathsf{italie}$
$\mathsf{france} \sqcap \mathsf{italie} \sqsubseteq \bot$	

$\mathsf{italie} \sqsubseteq \mathsf{lieu}$
$\mathsf{alpesFrancaises} \sqsubseteq \mathsf{lieuAvecMontagne}$
$\mathsf{alpesItaliennes} \sqsubseteq \mathsf{lieuAvecMontagne}$

Soit la requête normalisée :
$Q : \mathsf{q}(H) : \quad \mathsf{hôtel}(H) \wedge \mathsf{situationAssocié}(H, L)$
$\wedge(\mathsf{france} \sqcap \mathsf{lieuAvecMontagne} \sqcap \mathsf{ville})(L).$

Elle exprime la recherche d'hôtels situés en France, dans des villes de lieux montagneux.

Soit l'ensemble de vues disponibles $\mathcal{V} = \{\mathsf{v}_0, \mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3, \mathsf{v}_4\}$ dont les définitions sont :

$\mathit{def}(\mathsf{v}_0) = \mathsf{hôtelDeLuxe}$	$\mathit{def}(\mathsf{v}_3) = \mathsf{alpesFrancaises}$
$\mathit{def}(\mathsf{v}_1) = \mathsf{situationAssocie}$	$\mathit{def}(\mathsf{v}_4) = \mathsf{alpesItaliennes}$
$\mathit{def}(\mathsf{v}_2) = (\mathsf{ville} \sqcap \mathsf{régionPACA})$	

v_0 permet d'obtenir des hôtels de luxe ; v_1 permet d'obtenir des couples (a, b) d'instances reliées par le rôle situationAssocié ; v_2 permet d'obtenir des villes de la région Provence-Alpes-Côte-d'Azur ; v_3 permet d'obtenir des lieux des alpes françaises ; v_4 permet d'obtenir des lieux des alpes italiennes.

La requête $\mathsf{q}_{\mathcal{V}}(H) : \mathsf{v}_0(H) \wedge \mathsf{v}_1(H, L) \wedge \mathsf{v}_2(L) \wedge \mathsf{v}_3(L)$ est un plan de requête de Q. En effet, son expansion est :
$\mathsf{hôtelDeLuxe}(H) \wedge \mathsf{situationAssocie}(H, L) \wedge (\mathsf{ville} \sqcap \mathsf{régionPACA})(L) \wedge \mathsf{alpesFrancaises}(L)$.
D'une part, elle est satisfiable. D'autre part, comme hôtelDeLuxe est subsumé par hôtel, ville $\sqcap$ régionPACA est subsumé par ville, et alpesFrancaises est subsumé par lieuAvecMontagne, elle implique logiquement (modulo la terminologie) le corps de la requête q.
Ce plan de requête permet d'obtenir des hôtels de luxe situés dans les Alpes et dans des villes de la région PACA : ce sont bien des hôtels situés en France,

dans des villes de lieux montagneux.

En revanche, la requête $\mathsf{q}'_{\mathcal{V}}(H) : \mathsf{v}_0(H) \wedge \mathsf{v}_1(H, L) \wedge \mathsf{v}_2(L) \wedge \mathsf{v}_4(L)$ n'est pas un plan de requête de Q car son expansion :
$\mathsf{hôtelDeLuxe}(H) \wedge \mathsf{situationAssocie}(H, L) \wedge (\mathsf{ville} \sqcap \mathsf{régionPACA})(L) \wedge \mathsf{alpesItaliennes}(L)$
est insatisfiable. En effet, alpesItaliennes étant subsumé par italie, et ville $\sqcap$ régionPACA étant subsumé par régionPACA, lui-même subsumé par france, il découle logiquement de l'expansion de $\mathsf{q}'_{\mathcal{V}}$ la conjonction $\mathsf{france}(L) \wedge \mathsf{italie}(L)$, et sa normalisation $(\mathsf{france} \sqcap \mathsf{italie})(L)$. Or, le concept france$\sqcap$italie est insatifisfiable modulo la terminologie. □

4.2 Principales étapes du calcul de plans de requêtes

Le calcul des plans de requêtes pour une requête q en fonction d'un ensemble de vues $\mathcal{V}$ s'effectue en plusieurs étapes :

Etape 1 : Vérification de la satisfiabilité de la requête normalisée.
La vérification consiste à :

(i) faire appel à ONTOCLASS afin de vérifier que chaque concept C apparaissant dans un atome-concept $C(u)$ de la requête est satisfiable modulo la terminologie,

(ii) saturer en chaînage-avant le corps de la requête en appliquant itérativement toutes les règles de la composante déductive jusqu'à ce que $\perp$ soit généré ou qu'aucun nouvel atome ne puisse être produit par application de règles.

Si un concept de la requête est détecté insatisfiable ou si $\perp$ a été inféré, cela signifie qu'une contrainte d'intégrité a été déclenchée et que la requête est insatisfiable. Dans ce cas, le calcul des plans de requêtes n'est pas lancé et la requête est transmise au module d'affinement de requêtes décrit dans la section suivante.

Etape 2 : dépliement de la requête.
L'opération de dépliement de la requête s'effectue par chaînage-arrière et consiste à remplacer chaque atome ordinaire apparaissant dans le corps de la requête q par le corps de chaque règle le définissant :

- pour chaque atome ordinaire $p(\bar{X}')$ du corps de la requête q :

- pour chaque règle qui, à un renommage des variables près, est de la forme :

$$p(\bar{X}') : q_1(\bar{X}'_1, \bar{Z}'_1) \wedge \ldots \wedge q_p(\bar{X}'_p, \bar{Z}'_p) :$$

- créer une requête q' obtenue à partir de q en remplaçant dans le corps de q

l'atome $p(\bar{X}')$ par la conjonction d'atomes $q_1(\bar{X}'_1, \bar{U}_1) \wedge \ldots \wedge q_p(\bar{X}'_p, \bar{U}_p)$,

où $\bar{U}_1, \ldots, \bar{U}_p$ sont des vecteurs de nouvelles variables.

- si aucune règle concluant sur $\perp$ n'est applicable sur le corps de q', déplier la requête q'.

Le résultat est un ensemble de requêtes conjonctives satisfiables appelé *déplié* de la requête initiale.

Exemple 18
Soit la requête :

$$\mathsf{q}(H, S) : \quad \mathsf{hôtel}(H) \wedge \mathsf{situationAssocié}(H, L) \wedge (\mathsf{france} \sqcap \mathsf{ville})(L) \wedge \mathsf{loisirPraticable}(L, S).$$

Elle exprime la recherche de couples d'hôtel et de loisir tels que l'hôtel est situé dans une ville de France où le loisir peut être pratiqué.

En considérant les définitions de loisirPraticable données dans l'exemple 8 (page 19), les dépliés de la requête sont :

$$\begin{array}{ll}
\mathsf{q}_1(H, S) : & \mathsf{hôtel}(H) \wedge \mathsf{situationAssocié}(H, L) \wedge (\mathsf{france} \sqcap \mathsf{ville})(L) \\
& \wedge \mathsf{lieuAvecMontagne}(L) \wedge \mathsf{sportDeMontagne}(S); \\
\mathsf{q}_2(H, S) : & \mathsf{hôtel}(H) \wedge \mathsf{situationAssocié}(H, L) \wedge (\mathsf{france} \sqcap \mathsf{ville})(L) \\
& \wedge \mathsf{lieuAvecPlage}(L) \wedge \mathsf{sportNautique}(S); \\
\mathsf{q}_3(H, S) : & \mathsf{hôtel}(H) \wedge \mathsf{situationAssocié}(H, L) \wedge (\mathsf{france} \sqcap \mathsf{ville})(L) \\
& \wedge \mathsf{lieuAvecPiscine}(L) \wedge \mathsf{natation}(S).
\end{array}$$

□

Etape 3 : Réécriture atome par atome de chaque requête conjonctive du déplié.

Une étape de réécriture d'une requête conjonctive consiste à créer des réécritures partielles obtenues en remplaçant un atome du corps de la requête

par sa réécriture en terme de vues. Le processus de réécriture est itéré sur les réécritures partielles tant qu'elles contiennent des atomes à réécrire. Le processus s'arrête donc quand toutes les réécritures partielles ne contiennent que des atomes-vues. A chaque étape de réécriture, on vérifie que le résultat produit est satisfiable. Si ce n'est pas le cas, l'algorithme de réécriture s'arrête. Si, pour une requête donnée, aucune réécriture n'a été obtenue (i.e., toutes les réécritures partielles sont insatisfiables), le module d'affinement de requêtes prend le relais.

La réécriture d'un atome-rôle ou d'un atome ordinaire est triviale : le résultat obtenu est l'ensemble des atomes-vues de la forme $v(X_1, X_2)$ (ou $v(X_1, \ldots, X_n)$) tels que $v \in \mathcal{V}$ et $def(v) = \mathsf{r}$ (ou $def(v) = \mathsf{p}$).

En revanche, la réécriture d'un atome-concept est plus complexe. Nous commençons par l'illustrer sur un exemple. Nous présentons ensuite les règles de réécriture qui doivent être mises en oeuvre pour obtenir l'ensemble de toutes les réécritures possibles.

Exemple 19
Soient l'ensemble $\{\mathsf{Déjeuner}, \mathsf{Lieu}, \mathsf{VilleEU}, \mathsf{Vol}\}$ de concepts, et l'ensemble $\{\mathsf{Destination}, \mathsf{Escale}, \mathsf{MoyenDAccès}, \mathsf{Prestation}, \mathsf{Repas}, \mathsf{SéanceCinéma}\}$ de rôles.

Considérons la requête q dont la définition est :

$$\mathsf{C}(X) : \quad (\mathsf{Vol} \sqcap (\forall \mathsf{Destination}\ \mathsf{VilleEU}) \sqcap (\geq 1\ \mathsf{Escale}) \sqcap (\forall \mathsf{Prestation}\ (\geq 1\ \mathsf{SéanceCinéma})))(X).$$

Elle exprime la recherche de vols à destination des États Unis, ayant au moins une escale et dont toutes les prestations (si elles existent) consistent en au moins une séance de cinéma.

Pour réécrire cette requête, nous disposons de l'ensemble de vues $\mathcal{V} = \{\mathsf{v}_0, \mathsf{v}_1, \mathsf{v}_2, \mathsf{v}_3, \mathsf{v}_4, \mathsf{v}_5, \mathsf{v}_6, \mathsf{v}_7, \mathsf{v}_8, \mathsf{v}_9\}$ dont les définitions sont :

$\mathit{def}(\mathsf{v}_0) = \mathsf{Vol}$
$\mathit{def}(\mathsf{v}_1) = (\mathsf{Lieu} \sqcap (\forall\, \mathsf{MoyenDAccès}\, \mathsf{Vol})\,)$
$\mathit{def}(\mathsf{v}_2) = \mathsf{MoyenDAccès}$
$\mathit{def}(\mathsf{v}_3) = (\leq\, 1\, \mathsf{Destination})$
$\mathit{def}(\mathsf{v}_4) = \mathsf{Destination}$
$\mathit{def}(\mathsf{v}_5) = \mathsf{VilleEU}$
$\mathit{def}(\mathsf{v}_6) = \mathsf{Escale}$
$\mathit{def}(\mathsf{v}_7) = (\forall\, \mathsf{Prestation}\, (\forall\, \mathsf{Repas}\, \mathsf{Déjeuner}))$
$\mathit{def}(\mathsf{v}_8) = (\forall\, \mathsf{Prestation}\, (\forall\, \mathsf{Repas}\, \neg\mathsf{Déjeuner}))$
$\mathit{def}(\mathsf{v}_9) = (\forall\, \mathsf{Prestation}\ (\geq\, 2\, \mathsf{Repas}))$

Dans le cadre d'un médiateur, ces vues décrivent, en termes du schéma global, les réponses à des requêtes obtenues à partir de sources de données : v_0 permet d'obtenir des vols ; v_1 permet d'obtenir des lieux qui s'ils sont accessibles, ne le sont que par voie aérienne ; v_2 permet d'obtenir des couples (a,b) d'instances reliées par le rôle MoyenDAccès ; v_3 permet d'obtenir des instances ayant au plus une destination ; v_4 permet d'obtenir des couples d'instances reliées par le rôle Destination ; v_5 permet d'obtenir des villes des États Unis ; v_6 permet d'obtenir des couples d'instances reliées par le rôle Escale ; v_7 (respectivement v_8) permet d'obtenir des instances dont toutes les prestations, s'il en existe et si elles offrent des repas, n'offrent que des déjeuners (respectivement que des repas qui ne sont pas des déjeuners) ; v_9 permet d'obtenir des instances dont les prestations, s'il en existe, consistent en au moins deux repas.

Pour calculer les réécritures de la requête, l'algorithme considère chaque composant de la conjonction, réécrit successivement $\mathsf{Vol}(X)$, $(\forall\, \mathsf{Destination}\ \mathsf{VilleEU})(X)$, $(\geq\, 1\ \mathsf{Escale})(X)$, $(\forall\, \mathsf{Prestation}\ \ (\geq\, 1\ \mathsf{SéanceCinéma}))\ (X)$, et combine conjonctivement leurs réécritures.

Les réécritures des atomes $\mathsf{Vol}(X)$ et $(\geq\, 1\ \mathsf{Escale})(X)$ s'obtiennent de manière relativement directe à partir des sources. Les réécritures de $\mathsf{Vol}(X)$ sont d'une part $\mathsf{v}_0(X)$ et d'autre part $\mathsf{v}_1(Y_1) \wedge \mathsf{v}_2(Y_1, X)$, où Y_1 est une variable existentielle. L'unique réécriture de $(\geq 1\, \mathsf{Escale})(X)$ a pour définition : $\mathsf{v}_6(X, Y_3)$, où Y_3 est une variable existentielle.

Les réécritures des atomes $(\forall\mathsf{Destination}\ \mathsf{VilleEU})(X)$ et $(\forall\mathsf{Prestation}\ (\geq 1\ \mathsf{SéanceCinéma}))(X)$ mettent en évidence deux points subtils de la réécriture d'atomes-concepts, qui sont nécessaires pour garantir sa complétude.

L'unique réécriture de $(\forall \mathsf{Destination}\ \mathsf{VilleEU})(X)$ a pour définition : $\mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2) \wedge \mathsf{v}_5(Y_2)$, où Y_2 est une variable existentielle. $\mathsf{v}_3(X)$ désigne les X qui ont au plus une destination. La conjonction $\mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2)$ exhibe les X qui ont donc exactement une destination Y_2. Enfin, la conjonction $\mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2) \wedge \mathsf{v}_5(Y_2)$ représente les X ayant exactement une destination Y_2 qui est une ville des États Unis : il en découle logiquement que toutes les destinations possibles de ces X sont des villes des États Unis, et que donc $\mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2) \wedge \mathsf{v}_5(Y_2)$ est une réécriture de $(\forall \mathsf{Destination}\ \mathsf{VilleEU})(X)$. L'unique réécriture de $(\forall \mathsf{Prestation}\ (\geq 1\ \mathsf{SéanceCinéma}))(X)$ a pour définition : $\mathsf{v}_7(X) \wedge \mathsf{v}_8(X) \wedge \mathsf{v}_9(X)$. Elle s'explique de la façon suivante. Le rôle SéanceCinéma n'apparaissant pas dans les définitions des vues de $\mathcal{V}$, la seule réécriture possible consiste à inférer à partir des vues : $(\forall \mathsf{Prestation}\ \bot)$, c'est à dire que le nombre de prestations est nul ($(\forall \mathsf{Prestation}\ \bot) \equiv (\leq 0\ \mathsf{Prestation})$). L'algorithme construit donc la conjonction $\mathsf{v}_7(X) \wedge \mathsf{v}_8(X)$ dont l'expansion est $(\forall \mathsf{Prestation}\ (\forall \mathsf{Repas}\ \bot))(X)$: les X dont aucune prestation ne propose de repas. Puis, il construit la conjonction $\mathsf{v}_7(X) \wedge \mathsf{v}_8(X) \wedge \mathsf{v}_9(X)$ dont l'expansion $(\forall \mathsf{Prestation}\ (\forall \mathsf{Repas}\ \mathsf{Déjeuner}))(X) \wedge (\forall \mathsf{Prestation}\ (\forall \mathsf{Repas}\ \neg\mathsf{Déjeuner}))(X) \wedge (\forall \mathsf{Prestation}\ (\geq 2\ \mathsf{Repas}))(X)$ est équivalente à $(\forall \mathsf{Prestation}\ \bot)(X)$, car elle désigne les X dont les prestations sont à la fois de ne pas proposer de repas et de proposer deux repas.

Finalement, par conjonction des différentes réécritures, l'algorithme construit les réécritures q_v^1 et q_v^2 pour la requête q :

$$\begin{aligned}
\mathsf{q}_v^1(X) : \quad & \mathsf{v}_0(X) \wedge \mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2) \wedge \mathsf{v}_5(Y_2) \wedge \mathsf{v}_6(X, Y_3) \wedge \mathsf{v}_7(X) \\
& \wedge \mathsf{v}_8(X) \wedge \mathsf{v}_9(X) \\
\mathsf{q}_v^2(X) : \quad & \mathsf{v}_1(Y_1) \wedge \mathsf{v}_2(Y_1, X) \wedge \mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2) \wedge \mathsf{v}_5(Y_2) \\
& \wedge \mathsf{v}_6(X, Y_3) \wedge \mathsf{v}_7(X) \wedge\ \mathsf{v}_8(X) \wedge \mathsf{v}_9(X).
\end{aligned}$$

□

Cet exemple, bien que peu réaliste, permet d'illustrer la variété des interactions des différents constructeurs et la diversité des réécritures qui peuvent en résulter. Il donne une certaine intuition de la complexité potentielle d'un algorithme de réécriture **complet**.

Dans [20], nous décrivons un algorithme de réécriture de requêtes conjonctives de CARIN-$\mathcal{ALN}$, dont nous prouvons la terminaison, la correction et la complétude, et dont nous étudions la complexité en temps dans le cas le pire. Il est fondé sur la mise en oeuvre des règles de réécriture suivantes selon une stratégie qui garantit sa terminaison.

(1) Réécriture d'un atome-rôle $\mathsf{r}(U_0, U_1)$ ***ou ordinaire*** $\mathsf{p}(U_0, \ldots, U_n)$ ***:***

La réécriture de ce type d'atome est une condition d'arrêt de l'algorithme. Elle consiste à construire des atome-vues $\mathsf{v}(U_0, U_1)$ ou $\mathsf{v}(U_0, \ldots, U_n)$ à partir de vues dont la définition est r ou p.

(2) Réécriture d'un atome-concept de la forme $(\bigsqcap_{i=1}^{n} \mathsf{C}_i)(U_0)$ ***:***

La réécriture de ce type d'atome s'effectue en combinant les réécritures de chacun des atome-concepts $\mathsf{C}_i(U_0)$. Cette règle est fondée sur l'implication logique suivante (en fait une équivalence) :

$$(1)\ \bigwedge_{i=1}^{n} \mathsf{C}_i(U_0) \models (\bigsqcap_{i=1}^{n} \mathsf{C}_i)(U_0).$$

(3) Réécriture d'un atome-concept simple $\mathsf{C}(U_0)$ ***(c-à-d*** C ***n'est pas une conjonction) :***

La réécriture de ce type d'atome est une condition d'arrêt de l'algorithme. Cette règle est fondée sur l'implication logique suivante où $\mathsf{D} \preceq_{\mathcal{T}} \mathsf{C}$:

$$(2)\ (\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_n\ \mathsf{D}))(U_n) \wedge \bigwedge_{i=1}^{n} \mathsf{r}_i(U_{n+1-i}, U_{n-i}) \models \mathsf{C}(U_0),$$

Exemple 20
Dans l'exemple 19, les deux réécritures possibles de $\mathsf{Vol}(X)$ sont trouvées grâce à cette règle. La première, $\mathsf{v}_0(X)$, a pour expansion $\mathsf{Vol}(X)$. Dans ce cas particulier, la chaîne $(\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_n\ ))$ est vide ($n = 0$) et $\mathsf{C} = \mathsf{Vol}$. La seconde, $\mathsf{v}_1(Y_1) \wedge \mathsf{v}_2(Y_1, X)$, a pour expansion $(\mathsf{Lieu} \sqcap (\forall \mathsf{MoyenDAccès}\ \mathsf{Vol}))(Y_1)$ $\wedge \mathsf{MoyenDAccès}(Y_1, X)$. □

(4) Réécriture d'un atome-concept de la forme $(\geq n\ \mathsf{r})(U_0)$ ***:***

La réécriture de ce type d'atome s'effectue récursivement en combinant les réécritures de chacun des atome-rôles de la partie gauche de l'implication

logique :

$$(3)\ \bigwedge_{i=1}^{n} \mathsf{r}(U_0, U_i) \wedge \bigwedge_{i=1}^{n} \bigwedge_{\substack{j=i+1 \\ i \neq j}}^{n} U_i \neq U_j \models (\geq n\ \mathsf{r})(U_0).$$

Exemple 21
Dans l'exemple 19, la réécriture de $(\geq 1\ \mathsf{Escale})(X)$ est trouvée grâce à cette règle. Cette réécriture $\mathsf{v}_6(X, Y_3)$ a pour expansion $\mathsf{Escale}(X, Y_3)$. □

(5) Réécriture d'un atome-concept de la forme $(\forall \mathsf{r}\ \mathsf{C})(U_0)$***:***

La réécriture de ce type d'atome s'effectue récursivement en combinant les réécritures de chacun des atome-concepts et atome-rôles de la partie gauche de l'implication logique :

$$(4)\ (\leq n\ \mathsf{r})(U_0) \wedge \bigwedge_{i=1}^{n} (\mathsf{r}(U_0, U_i) \wedge \mathsf{C}(U_i)) \wedge \bigwedge_{i=1}^{n} \bigwedge_{\substack{j=i+1 \\ i \neq j}}^{n} U_i \neq U_j \models (\forall \mathsf{r}\ \mathsf{C})(U_0).$$

Exemple 22
Dans l'exemple 19, la réécriture de $(\forall \mathsf{Destination}\ \mathsf{VilleEU})(X)$ est trouvée grâce à cette règle. Cette réécriture, $\mathsf{v}_3(X) \wedge \mathsf{v}_4(X, Y_2) \wedge \mathsf{v}_5(Y_2)$, a pour expansion $(\leq 1\ \mathsf{Destination})(X) \wedge \mathsf{Destination}(X, Y_2) \wedge \mathsf{VilleEU}(Y_2)$. □

(6) Réécriture d'un atome-concept de la forme $(\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_n\ \mathsf{C}))$ ***où*** $\mathsf{C} \neq (\forall \mathsf{r}\ \mathsf{D})$***:***

Certaines réécritures de ce type d'atome ne peuvent pas être trouvées par les règles de réécriture présentées ci-dessus, qui ne couvrent pas les implications logiques suivantes pourtant valides pour un concept C quelconque :

$$(\forall \mathsf{r}_1\ \bot)(X) \models \ldots \models (\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_{k-1}\ \bot))(X) \models (\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_k\ \mathsf{C}))(X).$$

Trouver toutes les réécritures d'un atome-concept $(\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_n\ \mathsf{C}))(X)$ nécessite donc de calculer toutes les réécritures des $(\forall \mathsf{r}_1\ \cdots (\forall \mathsf{r}_i\ \bot))(X)$,

avec $0 < i < n$.

Ces réécritures sont obtenues à partir des ensembles minimalement insatisfiables de descriptions de concepts pouvant être construits à l'aide de définitions de vues. Un ensemble $\mathcal{E}$ de descriptions de concepts est *minimalement insatisfiable* ssi pour tout sous ensemble non vide ε de $\mathcal{E}$, $\mathcal{E} \setminus \varepsilon$ est satisfiable.

Dans $\mathcal{ALN}$, les ensembles minimalement insatisfiables peuvent être explicités. Ils sont nécessairement de la forme :

- $\{\mathsf{A}, \neg\mathsf{A}\}$, où A est un concept de base quelconque ;
- $\{(\geq p\ \mathsf{r}), (\leq q\ \mathsf{r})\}$, où $p > q$ et r est un rôle quelconque ;
- $\{(\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_k\ \mathsf{A})), (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_k\ \neg\mathsf{A})), (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_{k-1}\ (\geq 1\ \mathsf{r}_k))), \ldots, (\forall \mathsf{r}_1\ (\geq 1\ \mathsf{r}_2)), (\geq 1\ \mathsf{r}_1)\}$,
  où A est un concept de base quelconque et les r_i sont des rôles quelconques ;
- $\{(\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_k\ (\geq p\ \mathsf{r}))), (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_k\ (\leq q\ \mathsf{r}))), (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_{k-1}\ (\geq 1\ \mathsf{r}_k))), \cdots, (\forall \mathsf{r}_1\ (\geq 1\ \mathsf{r}_2)), (\geq 1\ \mathsf{r}_1)\}$, où $p > q$ et r et les r_i sont des rôles quelconques ;
- $\{(\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_k\ \bot)), (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_{k-1}\ (\geq 1\ \mathsf{r}_k))), \cdots (\forall \mathsf{r}_1\ (\geq 1\ \mathsf{r}_2)), (\geq 1\ \mathsf{r}_1)\}$, où les r_i sont des rôles quelconques.

La règle de réécriture utilisée est la suivante ; elle s'applique pour un concept C quelconque, et pour tout $j \in [1..n]$ et tout ensemble minimalement insatisfiable $\{\mathsf{C}_1, \ldots, \mathsf{C}_k\}$, où les $\mathsf{C}_1, \ldots, \mathsf{C}_k$ sont des descriptions de concepts apparaissant dans les définitions des vues :

$$(5)\ \bigwedge_{i=1}^{k} (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_j\ \mathsf{C}_i))(X) \models (\forall \mathsf{r}_1 \cdots (\forall \mathsf{r}_n\ \mathsf{C}))(X).$$

Exemple 23
Dans l'exemple 19, la réécriture de $(\forall \mathsf{Prestation}\ (\geq 1\ \mathsf{SéanceCinéma}))(X)$ est trouvée grâce à cette règle, appliquée avec $\mathsf{C} = (\geq 1\ \mathsf{SéanceCinéma})$ et

$\mathsf{r}_1 = \mathsf{Prestation}$. Cette réécriture, $\mathsf{v}_7(X) \wedge \mathsf{v}_8(X) \wedge \mathsf{v}_9(X)$, a pour expansion :

$$\left.\begin{array}{l}(\forall \mathsf{Prestation}\ (\forall \mathsf{Repas}\ \mathsf{Déjeuner}))(X) \\ \wedge\ (\forall \mathsf{Prestation}\ (\forall \mathsf{Repas}\ \neg\mathsf{Déjeuner}))(X) \\ \wedge\ (\forall \mathsf{Prestation}\ (\geq 2\ \mathsf{Repas}))(X)\end{array}\right\} \equiv (\forall \mathsf{Prestation}\ \bot)(X).$$

Dans ce cas, l'ensemble minimalement insatisfiable issu des vues est :
$\{(\forall \mathsf{Repas}\ \mathsf{Déjeuner}), (\forall \mathsf{Repas}\ \neg\mathsf{Déjeuner}), (\geq 2\ \mathsf{Repas})\}$. □

Nous avons établi dans [20] une borne supérieure de la complexité en temps dans le pire des cas de l'algorithme de réécriture implantant l'ensemble des règles de réécritures précédentes. Elle est :

- au plus polynomiale en fonction du nombre de vues et de la longueur maximale de leur définition (la longueur d'une expression de concept $C_1 \sqcap \cdots \sqcap C_l$ est le nombre l de ses composants conjonctifs) ;
- au plus exponentielle en fonction de la taille de la requête ;
- au plus en $O(p^{p^2})$ où p est la profondeur maximale d'imbrication de constructeurs $\forall$ apparaissant dans la définition des vues, et en $O(n^n)$ où n est le plus grand entier naturel pouvant apparaître dans une restriction de cardinalité supérieure au sein d'une vue.

Les résultats de complexité présentés ci-dessus, bien qu'ils soient peu révélateurs de l'efficacité de l'algorithme de réécriture dans le cadre d'applications réelles car ce sont des *majorants de complexité dans le pire des cas*, permettent d'exhiber différents paramètres auxquels l'algorithme est plus ou moins sensible.
Le paramètre dont la valeur est par nature élevée est le nombre de vues. L'analyse ci-dessus montre que l'algorithme est peu sensible à ce paramètre : sa complexité est au pire polynomiale en fonction de ce paramètre.
La taille de la requête, en fonction de laquelle la complexité de l'algorithme est exponentielle, est en général très petite dans la pratique. En effet, le nombre maximal d'imbrications de restrictions universelles est rarement supérieur à deux imbrications, car au-delà, il est difficile pour un utilisateur de donner une signification claire à une description de concept, même si celle-ci a une définition rigoureusement définie par la théorie des modèles. Le

nombre de conjonctions de la requête est également très petit car les utilisateurs posent spontanément des requêtes plutôt générales quitte à les raffiner (ce qui fait croître légèrement le nombre de conjonctions) s'ils obtiennent trop de réponses.
Les paramètres concernant les vues sont eux plus variables. La profondeur maximale (p) des descriptions de concepts définissant les vues est petite pour les mêmes raisons que ci-dessus. En revanche, selon les choix de modélisation, la longueur maximale (l) des descriptions de concepts définissant les vues ainsi que le plus grand entier naturel (n) apparaissant dans une restriction de cardinalité supérieure peuvent être grands. L'algorithme est peu sensible à la longueur maximale des définitions des vues, puisque sa complexité est polynomiale en l. Enfin, l'algorithme est sensible à n car sa complexité est en $O(n^n)$. Ce facteur étant directement lié aux choix de modélisation des vues, il doit donc impérativement être pris en compte dans la phase de modélisation des vues pour que le calcul des plans de requêtes puisse être performant dans la pratique.

Dans le cadre de PICSEL, nous avons fait le choix de ne pas implanter les parties des algorithmes sources d'une grande complexité ou concernant la réécriture de requêtes très particulières rarement formulées par des utilisateurs. Ainsi, la règle de réécriture (6) basée sur la recherche, dans les vues, d'ensembles de descriptions de concepts minimalement insatisfiables n'a pas été implémentée. L'inconvénient théorique est que l'algorithme implanté n'est pas complet en termes des réécritures qu'il produit. Toutefois, cette incomplétude est très précisément caractérisée et peut disparaître en réintroduisant le codage de la règle de réécriture correspondante.

5 AFFINEMENT DE REQUÊTES DANS PICSEL

Cette partie présente l'état actuel de la composante *affinement de requêtes* du médiateur, dont l'objectif est d'aider l'utilisateur à reformuler sa requête lorsque celle-ci n'obtient pas de réponses. Cette composante entre dans la constitution d'un module de dialogue coopératif entre le médiateur et l'utilisateur.

Une requête peut ne pas avoir de réponse, soit parce que l'utilisateur l'a mal posée et qu'elle viole des contraintes d'intégrité du domaine, soit parce qu'elle ne peut être mise en correspondance avec les sources d'informations

existantes[6].

Le module d'affinement intervient en liaison avec l'algorithme de calcul des plans de requêtes, (i) quand l'étape de vérification de satisfiabilité de la requête conclut que celle-ci est insatisfiable, (4.2, étape 1), (ii) quand toutes les réécritures partielles calculées dans l'étape 3 sont insatisfiables (requête non réécrivable à partir des sources disponibles).

Quand la requête initiale est insatisfiable, la première étape de l'affinement consiste à identifier les plus petits sous-ensembles de connaissances du domaine et de la requête qui sont à l'origine de l'échec. Pour cela, il faut extraire des règles et des contraintes, celles qui, avec le corps de la requête de l'utilisateur, impliquent logiquement le concept $\perp$, et extraire les atomes de la requête qui sont à l'origine de cette insatisfiabilité. Ces informations sont regroupées dans des triplets, appelés conflits. Parmi ces conflits, on distingue les conflits *pertinents*, qui regroupent les ensembles minimaux de connaissances et d'atomes permettant d'inférer $\perp$. Ces conflits identifient les concepts de la requête responsables de l'insatisfiabilité. En les remplaçant par des concepts plus généraux (au sens de la subsomption de l'ontologie), on obtient une requête satisfiable, sémantiquement proche de la requête intiale, appelée *réparation*, qui peut être proposée à l'utilisateur.

Le module d'affinement de requêtes comporte deux noyaux. Le premier noyau concerne le cas (i) des requêtes insatisfiables, et comprend un algorithme de détection de conflits et un algorithme de calcul des réparations minimales. Il fait l'objet des sous-sections 5.1 à 5.4. Le deuxième noyau concerne le cas (ii) des requêtes non réécrivables par manque de sources. Il est composé d'une base de requêtes prédéfinies construites automatiquement à partir de la description des sources, et d'un algorithme de sélection, dans cette base de requêtes, d'une requête proche de celle de l'utilisateur. Ce noyau est présenté en sous-section 5.5 et est actuellement en cours d'achèvement.

5.1 Conflits d'une requête

Dans ce module, on travaille sur une compilation de la composante terminologique $\mathcal{T}$ construite par le classifieur ONTOCLASS. Cette compilation traduit la relation de subsomption $\mathcal{C} \preceq \mathcal{C}'$ entre deux concepts de la hiérarchie

6. On ne s'intéresse pas ici, à une troisième cause d'échec possible, le cas où les informations supposées présentes dans la source ne sont pas disponibles.

de concepts de $\mathcal{T}$, par la règle $\mathcal{C}(x) \rightarrow \mathcal{C}'(x)$. On note $\mathcal{D}_h$ l'ensemble des règles traduisant ainsi la hiérarchie de concepts de la composante terminologique $\mathcal{T}$ de l'ontologie du domaine. De même, on traduit chaque contrainte d'exclusion $A_1 \sqcap A_2 \sqsubseteq \perp$ de la terminologie, par une contrainte d'intégrité $A_1(x) \wedge A_2(x) \rightarrow \perp$, et on note $\mathcal{C}_h$ l'ensemble des contraintes d'intégrité ainsi obtenues. L'ensemble des règles du domaine est ainsi $\mathcal{D} = \mathcal{D}_r \cup \mathcal{D}_h$, et l'ensemble des contraintes $\mathcal{C} = \mathcal{C}_r \cup \mathcal{C}_h$. On considèrera indifféremment le corps d'une requête comme un ensemble ou une conjonction d'atomes.

Définition 7 (conflit, cause)
Soit Q une requête et soit Q_1 un sous-ensemble d'atomes du corps de Q. Soit $\mathcal{D}_1$ un sous-ensemble de l'ensemble $\mathcal{D}$ des règles du domaine et $\mathcal{C}_1$ un sous-ensemble de l'ensemble $\mathcal{C}$ des contraintes du domaine. Un conflit *pour une requête Q est un triplet $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ tel que $Q_1, \mathcal{D}_1, \mathcal{C}_1 \models \perp$. ($\mathcal{D}_1, \mathcal{C}_1$) est appelé la* cause *du conflit.*

Plusieurs conflits peuvent co-exister pour une même requête. Pour les calculer tous, on utilise, en chaînage avant, une variante de la résolution générale, l'hyper-résolution positive [10] qui permet d'obtenir des preuves de $\perp$, en ne générant que des hyper-résolvants positifs. Ce calcul est donné dans [8].

Exemple 24
Supposons que la figure 2 présente la hiérarchie de concepts traduite sous forme d'un ensemble $\mathcal{D}$ de règles (les numéros des règles apparaissent sur les arcs) et que $\mathcal{C}$ contienne les contraintes suivantes :
c_1: lieuAvecPlage(x) $\wedge$ lieuSansPlage(x) $\rightarrow \perp$
c_2: lieuAuSoleil(x) $\wedge$ lieuAvecNeige(x) $\rightarrow \perp$
Soit la requête initiale :
$Q(x)$: Réunion(x) $\wedge$ Madère(x) $\wedge$ lieuAvecNeige(x)

On peut identifier les conflits suivants :
cf_1 ({Réunion(x), Madère (x)},{r_4,r_6,r_8,r_{11}},{c_1})
cf_2 ({Réunion(x), lieuAvecNeige(x) },{r_3},{c_2})
cf_3 ({Madère(x), lieuAvecNeige(x) },{r_5},{c_2})

Le conflit cf_1 s'interprète comme suit : à partir de l'atome Réunion(x), on peut déclencher les règles r_4 puis r_8 et dériver successivement IleAvecPlage(x) puis lieuAvecPlage(x). A partir de l'atome Madère(x), on

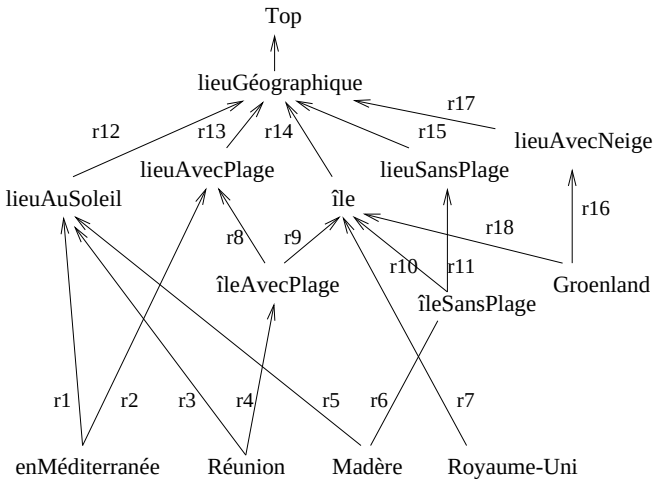

FIG. 2 – *Exemple de Hiérarchie sur les Lieux Géographiques*

peut déclencher les règles r_6 puis r_{11} et dériver IleSansPlage(x) puis lieuSansPlage(x). La présence simultanée de lieuAvecPlage(x) et lieuSansPlage(x) permet le déclenchement de la contrainte c_1 et la dérivation de $\perp$. □

L'ensemble des conflits intéressants doit permettre d'identifier tous les motifs d'insatisfiabilité de la requête, mais uniquement les éléments de la requête et de l'ontologie du domaine responsables (et indispensables) de cette insatisfiabilité. On s'intéresse plus particulièrement aux conflits qui sont minimaux, par rapport au sous-ensemble d'atomes du corps de la requête qui participe au conflit, (notion de *conflit Q_minimal*), et par rapport au nombre de règles et de contraintes impliquées, (notion de *cause minimale*). De tels conflits sont dits *pertinents*.

Définition 8 (conflit pertinent)
Un conflit $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *est* pertinent *si :*
(i) il est Q_minimal*, i.e., il n'existe pas de sous-ensemble strict* $Q_2 \subset Q_1$ *tel que* $Q_2, \mathcal{D}_1, \mathcal{C}_1 \models \perp$*,*

(ii) $(\mathcal{D}_1, \mathcal{C}_1)$ *est une* cause minimale *pour* Q_1*, i.e., il n'existe pas de cause* $(\mathcal{D}_2, \mathcal{C}_2) \subset (\mathcal{D}_1, \mathcal{C}_1)$[7] *telle que* $Q_1, \mathcal{D}_2, \mathcal{C}_2 \models \bot$.

Exemple 25
Les conflits cf_1, cf_2 et cf_3 sont des conflits pertinents vis-à-vis des connaissances présentées dans l'exemple 24. Si un atome, une règle ou une contrainte sont ôtés du conflit, il n'est plus possible de déduire le concept vide $\bot$.
En revanche, cf_4 ({Réunion(x), Madère(x), lieuAvecNeige(x) }, $\{r_3\}$,$\{c_2\}$) n'est pas pertinent car non Q_minimal et cf_5 ({Réunion(x), lieuAvecNeige(x) },$\{r_3\}$,$\{c_1,c_2\}$) n'est pas pertinent car ($\{r_3\}$,$\{c_1,c_2\}$) n'est pas une cause minimale pour {Réunion(x), lieuAvecNeige(x) }. □

Ces conflits peuvent être présentés à l'utilisateur comme une première et brutale explication de l'échec de la requête.

Une fois identifiés les conflits pertinents d'une requête, il faut les "supprimer". Pour chaque conflit, il suffit de considérer un de ses éléments et de le supprimer ou de le remplacer convenablement. La théorie du domaine étant supposée consistante, ce n'est pas elle qui est modifiée, mais les atomes du corps de la requête de l'utilisateur. L'objectif est alors de remplacer un (ou des) des atomes du conflit par une de ses généralisations, de façon à ce que les contraintes ne soient plus applicables pour inférer $\bot$.

5.2 Généralisation dans une hiérarchie de concepts

Pour rester le plus proche possible de la requête formulée par l'utilisateur, on fait le choix, pour chaque concept, de considérer l'ensemble de ses généralisants directs, plutôt qu'un seul. Ce choix n'est pas celui qui est fait habituellement. Il permet de prendre en compte les différents points de vue sur le concept. Comme la hiérarchie est structurée par un vocabulaire ciblé et partiellement redondant, on va retrouver, parmi les différents généralisants atteints, des concepts communs. De ce fait, le remplacement d'un concept par la conjonction de ses généralisants dans le corps de la requête de l'utilisateur ne fait pas exploser la taille de celle-ci.

7. $(\mathcal{D}_1, \mathcal{C}_1) \subset (\mathcal{D}_2, \mathcal{C}_2)$, si et seulement si ($\mathcal{D}_1 \subset \mathcal{D}_2$ et $\mathcal{C}_1 \subseteq \mathcal{C}_2$) ou ($\mathcal{D}_1 \subseteq \mathcal{D}_2$ et $\mathcal{C}_1 \subset \mathcal{C}_2$).

On définit tout d'abord les notions de généralisant direct et de généralisant d'un concept. La notion de généralisant définie ici dans un langage de règles, correspond à la notion de subsomption classiquement utilisée dans les langages de classes.

Définition 9 (généralisant)
Étant donné un ensemble $\mathcal{D}_h$ de règles définissant une hiérarchie, un généralisant direct *d'un concept $\mathcal{C}$ par rapport à $\mathcal{D}_h$ est un concept $\mathcal{C}'$ tel qu'il existe une règle $\mathcal{C}(x) \rightarrow \mathcal{C}'(x)$ dans $\mathcal{D}_h$. Top (noté $\top$) est l'unique généralisant direct des concepts $\mathcal{C}$ n'apparaissant jamais dans le corps d'une règle de $\mathcal{D}_h$. $\mathcal{C}'$ est un* généralisant *de $\mathcal{C}$ si $\mathcal{C}'$ est un généralisant direct de $\mathcal{C}$ ou s'il existe $\mathcal{C}''$ tel que $\mathcal{C}'$ est un généralisant direct de $\mathcal{C}''$ et $\mathcal{C}''$ est un généralisant de $\mathcal{C}$.*

Un concept peut avoir plusieurs généralisants directs.

Exemple 26
Dans la figure 2, les généralisants directs de Madère sont lieuAuSoleil et îleSansPlage. Ses autres généralisants sont île, lieuSansPlage et lieuGéographique. $\top$ est le généralisant direct[8] de lieuGéographique. □

On définit la notion de **généralisation** d'abord pour un atome-concept puis pour une conjonction d'atome-concepts.

Définition 10 (généralisation)
Soit $\mathcal{C}$ un concept et $\mathcal{C}_1$, $\mathcal{C}_2$,...,$\mathcal{C}_n$, l'ensemble de ses généralisants directs, la généralisation directe $\mathcal{G}_c(x)$ *de l'*atome-concept $\mathcal{C}(x)$ *est la conjonction construite à partir de ses généralisants directs $\mathcal{C}_1(x) \wedge \mathcal{C}_2(x) \wedge ... \wedge \mathcal{C}_n(x)$.*
Une généralisation directe $\mathcal{G}_d(x)$ *d'une* conjonction d'atome-concepts $\mathcal{C}_1(x) \wedge \ldots \wedge \mathcal{C}_{i-1}(x) \wedge \mathcal{C}_\mathbf{i}(\mathbf{x}) \wedge \mathcal{C}_{i+1}(x) \wedge \ldots \wedge \mathcal{C}_n(x)$ *est une conjonction de la forme $\mathcal{C}_1(x) \wedge \ldots \wedge \mathcal{C}_{i-1}(x) \wedge \mathcal{G}_\mathbf{ci}(\mathbf{x}) \wedge \mathcal{C}_{i+1}(x) \wedge \ldots \wedge \mathcal{C}_n(x), 1 \leq i \leq n$, où $\mathcal{G}_{ci}(x)$ est la généralisation directe de $\mathcal{C}_i(x)$.*
$\mathcal{G}(x)$ est une généralisation *d'une* conjonction d'atome-concepts $\mathcal{C}_{ac}(x)$ *si $\mathcal{G}(x)$ est une généralisation directe de $\mathcal{C}_{ac}(x)$ ou s'il existe une conjonction d'atome-concepts $\mathcal{G}'(x)$ telle que $\mathcal{G}(x)$ est une généralisation directe de $\mathcal{G}'(x)$ et $\mathcal{G}'(x)$ est une généralisation de $\mathcal{C}_{ac}(x)$.*

8. Dans une conjonction de littéraux, remplacer un atome par son généralisant $\top(x)$ revient à le supprimer.

Exemple 27
La généralisation directe de Madère(x) est lieuAuSoleil(x) $\wedge$ îleSansPlage(x). Une généralisation directe de la conjonction Madère(x) $\wedge$ réserveNaturelle(x) est lieuAuSoleil(x) $\wedge$ îleSansPlage(x) $\wedge$ réserveNaturelle(x). Une de ses généralisations est lieuAuSoleil(x) $\wedge$ lieuSansPlage(x) $\wedge$ réserve(x), si réserve(x) est une généralisation de réserveNaturelle(x). □

La propriété suivante, qui est immédiate, établit une caractérisation logique de la notion de généralisation.

Propriété : Soit $\mathcal{C}_{ac}(x)$ une conjonction de concepts, soit $\mathcal{D}_h$ un ensemble de règles définissant une hiérarchie de concepts et soit $\mathcal{G}(x)$ une généralisation de $\mathcal{C}_{ac}(x)$. On a : $\mathcal{C}_{ac}(x), \mathcal{D}_h \models \mathcal{G}(x)$.

5.3 Réparation de Requêtes

L'objectif de ce paragraphe est de caractériser les réparations qui doivent remplacer le corps de la requête de l'utilisateur $Q(x)$. Ces réparations doivent, bien évidemment, être satisfiables par rapport au domaine, mais également figurer parmi les plus petites généralisations satisfiables du corps de $Q(x)$ car l'on souhaite rester le plus près possible de $Q(x)$. L'ensemble des concepts apparaissant dans une réparation doit correspondre aux concepts les plus spécifiques de la hiérarchie permettant d'obtenir une requête satisfiable. Seules les réparations vérifiant ces critères seront présentées à l'utilisateur.

Définition 11 (Réparation-brute)
Soit $Q(\bar{X})$: $\mathcal{C}_{ac}(x) \wedge q(\bar{X})$ une requête insatisfiable, où $\mathcal{C}_{ac}(x)$ est une conjonction d'atome-concepts à généraliser, $x \in \bar{X}$, et $q(\bar{X})$ une conjonction d'atomes éventuellement vide. $\mathcal{R}(\bar{X})$ est une réparation-brute *de Q($\bar{X}$), si $\mathcal{R}(\bar{X})$ peut s'écrire sous la forme $\mathcal{C}'_{ac}(x) \wedge q(\bar{X})$, où $\mathcal{C}'_{ac}(x)$ est une généralisation de $\mathcal{C}_{ac}(x)$, et si $\mathcal{R}(\bar{X})$ est satisfiable.*

Exemple 28
On reprend la hiérarchie de la figure 2 et la contrainte
c_3 : Réunion(x) $\wedge$ Royaume-Uni(x) $\rightarrow \perp$.
Pour la requête $Q'(x)$: Réunion(x) $\wedge$ Royaume-Uni(x), on pose $\mathcal{C}_{ac}(x)$ = Réunion(x), q(x) = Royaume-Uni(x), et on choisit par exemple, parmi les

généralisations de Réunion(x), $\mathcal{C}'_{ac}(x)$ = lieuAuSoleil(x) $\wedge$ île(x).
La conjonction $\mathcal{R}(x)$ = lieuAuSoleil(x) $\wedge$ île(x) $\wedge$ Royaume-Uni(x) étant satisfiable, c'est une réparation brute. Les conjonctions Réunion(x) $\wedge$ île(x), lieuGéographique(x) $\wedge$ Royaume-Uni(x) et les deux conjonctions atomiques Réunion(x) et Royaume-Uni(x) sont également des réparations brutes de $Q'(x)$[9]. □

Un généralisant d'un concept est redondant par rapport à celui-ci : par exemple, le concept île est redondant par rapport au concept Royaume-Uni. En ôtant les généralisants redondants d'une réparation brute, on diminue sa taille et en la simplifiant, on facilite sa compréhension par l'utilisateur. On ne présente donc à l'utilisateur que des réparations ne contenant pas de redondances.

Définition 12 (Réparation)
Soit $Q(\bar{X})$ une requête insatisfiable. Une réparation-brute $\mathcal{R}(\bar{X})$ de $Q(\bar{X})$ est une réparation *de $Q(\bar{X})$ si elle ne contient pas deux concepts dont l'un soit un généralisant de l'autre.*

Exemple 29
Sur $Q'(x)$, en partant de la trentaine de réparations brutes de la requête, on obtient, parmi d'autres, les réparations suivantes :
lieuAuSoleil(x) $\wedge$ îleAvecPlage(x) $\wedge$ Royaume-Uni(x), lieuAuSoleil(x) $\wedge$ Royaume-Uni(x), lieuAuSoleil(x) $\wedge$ île(x), Réunion(x) et Royaume-Uni(x). □

La propriété suivante est immédiate et caractérise logiquement la notion de réparation.

Propriété : Soit $\mathcal{D}_h$ un ensemble de règles définissant une hiérarchie de concepts, soit $Q(\bar{X})$ une requête insatisfiable et soit $\mathcal{R}(\bar{X})$ une réparation de $Q(\bar{X})$. On a : corps($Q(\bar{X})$), $\mathcal{D}_h \models \mathcal{R}(\bar{X})$.

Cette définition reste encore assez large ; pour ne pas énumérer à l'utilisateur toutes les généralisations du corps de sa requête mais uniquement celles

9. Il en existe une trentaine, sans considérer les conjonctions qui contiennent plusieurs fois le même littéral.

qui en sont les plus proches, seules les **réparations minimales** lui seront présentées.

Définition 13 (réparation minimale)
$\mathcal{R}(\bar{X})$ *est une* réparation minimale *de* $Q(\bar{X})$ *si* $\mathcal{R}(\bar{X})$ *est une réparation de* $Q(\bar{X})$ *et s'il n'existe pas* $R'(\bar{X})$, *une autre réparation de* $Q(\bar{X})$, *telle que* $R'(\bar{X}), \mathcal{D}_h \models \mathcal{R}(\bar{X})$.

Exemple 30
Les réparations minimales de $Q'(x)$ sont les suivantes :
(i) lieuAuSoleil(x) $\wedge$ îleAvecPlage(x) $\wedge$ Royaume-Uni(x) et (ii) Réunion(x).
Dans (i), on a remplacé Réunion par ses deux généralisants directs compatibles avec Royaume-Uni, alors que, dans (ii), le seul généralisant de Royaume-Uni est redondant vis-à-vis de Réunion. □

Nous venons de définir ce que sont des réparations minimales. Nous présentons ci-dessous comment les calculer.

5.4 Calcul des réparations

Le calcul des réparations minimales nécessite de déterminer pour chaque conflit, des paires composées d'un **a**tome-concept et de **r**ègles de la hiérarchie consécutives, appelées **ar**-paires. Celles-ci sont calculées à partir des arbres d'hyper-résolution positive obtenus lors du calcul des conflits. Elles correspondent, d'une part, aux atomes à généraliser dans la requête et, d'autre part, aux ensembles minimaux des règles de $\mathcal{D}_h$ qui doivent être dépassées pour résoudre le conflit.

Pour chaque conflit, il faut dépasser au moins une de ses **ar**-paires. Cependant, les **ar**-paires des différents conflits peuvent parfois être identiques ou incluses l'une dans l'autre. Il faut donc déterminer un sous-ensemble minimal d'**ar**-paires à traiter pour obtenir une **réparation minimale** de la requête, qui supprime **tous** les conflits.

Ce problème est similaire au problème du diagnostic basé sur la consistance, dans laquelle on se donne un modèle de bon fonctionnement d'un

système. Le problème de diagnostic consiste alors à déterminer les composants du système tels que, supposer qu'ils sont anormaux rétablit la consistance entre le bon comportement attendu du système et son comportement observé [29]. Par analogie, les composants défectueux sont ici les atomes de la requête. Cependant, en diagnostic, les composants ont seulement deux états possibles, normal ou anormal. Réparer le système consiste alors à changer les composants anormaux, c'est-à-dire, à les remplacer par des composants normaux équivalents. Dans le cas présent, plutôt que de supprimer un ou des atomes défectueux par conflit, on les remplace par une de leurs généralisations.

On s'est donc inspiré de l'algorithme de calcul des *minimal hitting sets* de Reiter [29], en l'adaptant, pour prendre en compte les remplacements par généralisation. [6] présente un algorithme de calcul de l'ensemble des réparations minimales de la requête de l'utilisateur, à partir des **ar**-paires. Dans cet algorithme, on définit la notion d'arbre OMR, dont les feuilles permettent de calculer toutes les réparations minimales, et rien qu'elles (résultat de correction et de complétude).

5.5 Requêtes non réécrivables à partir des sources disponibles

On s'intéresse maintenant à des requêtes satisfiables avec l'ontologie du domaine, mais qui ne peuvent être réécrites par le médiateur, dans le vocabulaire des sources disponibles. Dans le cas où la requête de l'utilisateur échoue par suite de l'absence de sources pertinentes, elle est remplacée par une requête proche, appelée **solution**, c'est-à-dire, une requête qui peut être réécrite dans le vocabulaire des sources disponibles, et qui est sémantiquement proche de la requête initiale [7].

Les solutions à apporter sont issues d'une base de requêtes prédéfinies, pour lesquelles on sait qu'il existe *a priori* des réponses (requêtes réécrivables). Pour initialiser la base, chaque requête prédéfinie est construite à partir d'atomes dont une même source peut fournir différentes instances. On s'assure ainsi d'avoir au moins cette source pour réécrire complètement la requête. Puis cette base initiale sera progressivement enrichie et mise à jour en tenant compte des requêtes fréquemment posées par les utilisateurs et dont les réponses semblent jugées satisfaisantes.

On donne dans [7] une méthode de calcul qui permet de déterminer si une requête prédéfinie est suffisamment proche de celle de l'utilisateur pour

pouvoir lui être présentée comme **solution**. Pour cela, on définit à partir de la hiérarchie de concepts de l'ontologie du domaine, une notion de proximité de concepts, puis une notion de proximité entre requêtes.

Il est à noter que le procédé employé pour initialiser la base de requêtes prédéfinies, en utilisant les sources, est spécifique de l'approche médiateur adoptée dans le projet PICSEL.

5.6 Autres affinements

On a vu comment procéder dans les cas où l'utilisateur formule mal sa requête, soit parce qu'il viole des contraintes d'intégrité du domaine, soit parce qu'il sort des compétences des sources disponibles. Dans ces deux cas, la requête n'aura pas de réponse et devra être remplacée par une **réparation** ou par une **solution**. À l'inverse, il y a des cas où l'utilisateur formule bien sa requête, mais où celle-ci obtient trop de réponses pour qu'il soit raisonnable de toutes les lui proposer.

En cas de surabondance de réponses, le rôle du module d'affinement de requêtes est alors de sélectionner par spécialisation de la requête initiale, un sous-ensemble de réponses pertinentes, en tenant compte des préférences de l'utilisateur. Ce point est en cours d'étude.

6 CONCLUSION

Nous avons présenté le projet PICSEL qui offre un environnement déclaratif de construction de médiateurs fondé sur l'utilisation d'un langage logique combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (la logique de description $\mathcal{ALN}$).

Dans le cadre d'une collaboration avec Degriftour, nous avons développé une ontologie dans le domaine du tourisme. Nous avons montré au travers de cette application comment la modélisation de l'ontologie du domaine peut être guidée et contrainte par le choix du formalisme de représentation.

Nous avons fourni les règles de réécriture mises en oeuvre dans le calcul des plans de requêtes. L'implémentation de ce calcul a donné lieu au développement du prototype ONTOQUERY , intégrant le classifieur ONTOCLASS , qui ont tous deux fait l'objet d'une procédure de dépôt de logiciel par France Télécom R&D. Ces logiciels sont réutilisés dans le projet européen de commerce électronique MKBEEM [25] (Multilingual Knowledge Based European Electronic Market place) dirigé par France Télécom R&D.

Nous avons ensuite décrit le module d'affinement de requêtes pour lequel la maquette ONTOREPAIR a été développée.

Ce travail se poursuit actuellement dans le cadre de PICSEL2, toujours en collaboration avec France Télécom R&D. L'objectif de PICSEL2 est de montrer que l'approche médiateur, dont la faisabilité a été prouvée dans PICSEL, peut passer à l'échelle du Web. Pour cela, on envisage d'utiliser des techniques d'apprentissage pour acquérir de facon semi-automatique les concepts et leurs propriétés qui constituent le coeur de l'ontologie du domaine. Pour rendre cela possible, le choix fait dans PICSEL2 est de considérer que les sources de données à intégrer sont toutes décrites dans le même format (documents XML associés à des DTDs).

Le travail sur l'affinement de requêtes est poursuivi dans PICSEL2 pour déboucher sur un module de dialogue coopératif entre un médiateur et ses utilisateurs. En particulier, nous étudions les aides à apporter à un utilisateur qui obtient trop de réponses à sa requête.

RÉFÉRENCES

[1] B. Amann, I. Fundulaki et M. Scholl. Integrating Ontologies and Thesauri for RDF Schema Creation and Medata Querying. *International Journal of Digital Libraries*, 2000.

[2] S. Amer-Yahia, S. Cluet et C. Delobel. Bulk loading techniques for object databases and an application to relational data. In *VLDB 1998, Proceedings of 24th International Conference on Very Large Data Bases, NY, USA, August 1998*, Los Altos, CA 94022, USA, 1998. Morgan Kaufmann Publishers.

[3] Yigal Arens, Chung-Nan Hsu et Craig A. Knoblock. Query processing in the SIMS information mediator. In Austin Tate, éditeur, *Advanced Planning Technology*, pages 61–69. AAAI Press, Menlo Park, California, 1996.

[4] Yigal Arens et Craig A. Knoblock. SIMS: Retrieving and integrating information from multiple sources. In Peter Buneman et Sushil Jajodia, éditeurs, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 562–563, Washington, D.C., 26–28 Mai 1993.

[5] Domenico Beneventano, Sonia Bergamaschi, Silvana Castano, Alberto Corni, R. Guidetti, G. Malvezzi, Michele Melchiori et Maurizio Vin-

cini. Information integration: The MOMIS project demonstration. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter et Kyu-Young Whang, éditeurs, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt*, pages 611–614, Los Altos, CA 94022, USA, 2000. Morgan Kaufmann Publishers.

[6] A. Bidault, C. Froidevaux et B. Safar. Repairing queries in a mediator approach. In *14th European Conference on Artificial Intelligence*, pages 406–410, Berlin, 2000.

[7] A. Bidault, C. Froidevaux et B. Safar. Proximité entre requêtes dans un contexte médiateur. In *13ème Congrés Francophone de Reconnaissance des Formes et Intelligence Artificielle*, pages 653–662, Angers, 2002.

[8] Alain Bidault. Affinement de requêtes dans un médiateur. Université Paris XI – Orsay, 2002. Thèse de Doctorat.

[9] A. Borgida, R. J. Brachman, D. L. McGuinness et L. Alperin Resnick. CLASSIC: a structural data model for objects. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 18(2):58–67, Juin 1989.

[10] Chang et Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[11] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman et Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[12] Vassilis Christophides, Sophie Cluet et Jérôme Siméon. On Wrapping Query Languages and Efficient XML Integration. In *SIGMOD*, Dallas, Texas, Mai 2000.

[13] C-web project.
`http://cweb.inria.fr`.

[14] C. Delobel et M-C. Rousset. A uniform approach for querying large tree-structured data through a mediated schema. In *Proc. of the 2001 Int. Workshop on Foundations of Models for Information Integration*, September 2001.

[15] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi et Werner Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[16] Oren Etzioni et Daniel Weld. A Softbot-Based Interface to the Internet. *Communications of the ACM*, 37(7):72–76, 1994.

[17] Mary F. Fernandez, Daniela Florescu, Alon Y. Levy et Dan Suciu. Declarative Specification of Web Sites with Strudel. *VLDB journal*, 9(1):38–55, 2000.

[18] Marc Friedman et Daniel S. Weld. Efficiently executing information-gathering plans. In *15th International Joint Conference on Artificial Intelligence*, pages 785–791, Nagoya, Japan, 1997.

[19] Michael R. Genesereth, Arthur M. Keller et Oliver M. Duschka. Infomaster: an information integration system. In Joan M. Peckman, éditeur, *Proceedings, ACM SIGMOD International Conference on Management of Data: SIGMOD 1997: May 13–15, 1997, Tucson, Arizona, USA*, volume 26(2) of *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pages 539–542, New York, NY 10036, USA, 1997. ACM Press.

[20] François Goasdoué. Réécriture de requêtes en termes de vues dans CARIN et intégration d'informations. Université Paris XI – Orsay, novembre 2001. Thèse de Doctorat.

[21] François Goasdoué, Véronique Lattes et Marie-Christine Rousset. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. *International Journal of Cooperative Information Systems*, 9(4):383–401, décembre 2000.

[22] T. Kirk, A. Y. Levy, Y. Sagiv et D. Srivastava. The Information Manifold. In C. Knoblock et A. Levy, éditeurs, *Information Gathering from Heterogeneous, Distributed Environments*, AAAI Spring Symposium Series, Stanford University, Stanford, California, Mars 1995.

[23] R. MacGregor et R. Bates. The LOOM knowledge representation language. Technical Report ISI/RS-87-188, Information Science Institute, University of Southern California, Marina del Rey (CA), USA, 1987.

[24] E. Mena, Vipul Kashyap, Amit Sheth et A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In *4th Int. Conf. on Cooperative Information Systems*, pages 14–25, Bruessels, Belgium, 1996.

[25] MKBEEM project.
http://www.mkbeem.com.

[26] Y. Papakonstantinou, H. Garcia-molina et J. Widom. Object Exchange Across Heteregeneous Information Sources. In *ICDE Conf. on Management of Data*, 1995.

[27] Yannis Papakonstantinou, Hector Garcia-Molina et Jennifer Widom. Object exchange across heterogeneous information sources. In P. S. Yu et A. L. P. Chen, éditeurs, *11th Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.

[28] PICSEL project.
http://www.lri.fr/∼picsel/.

[29] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:406–410, 1987.

[30] C. Reynaud, J.-P. Sirot et D. Vodislav. Semantic Integration of XML Heterogeneous Data Sources. In *Proc. of the 2001 Int. Database Engineering & Applications Symposium*, July 2001.

[31] V. S. Subrahmanian, Sibel Adali, Anne Brink, Ross Emery, James J. Lu, Adil Rajput, Timothy J. Rogers, Robert Ross et Charles Ward. HERMES: A heterogeneous reasoning and mediator system. Technical report, University of Maryland, 1995.

[32] Gio Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.

[33] Xyleme.
http://www.xyleme.com.

[34] Lucie Xyleme. A dynamic warehouse for xml data of the web. *IEEE Data Engineering Bulletin*, 2001.