Réparation de requêtes posées à un médiateur

Repairing queries asked to a mediator

A. Bidault B. Safar
L.R.I., UMR 8623 C.N.R.S - Université Paris-Sud
Bâtiment 490, 91405, Orsay Cedex, France
{bidault, safar}@lri.fr

Résumé

Un médiateur est un système à base de connaissances qui regroupe, dans un cadre unificateur et homogène, les connaissances d'un domaine d'application et la description du contenu de différentes sources de données disponibles et pertinentes pour ce domaine. Dans cet article, nous travaillons sur les requêtes posées à un médiateur qui n'obtiennent pas de réponse. Tout d'abord, nous caractérisons en termes de conflits *les causes minimales de l'absence de réponse à une requête. Nous construisons ensuite son ensemble de* réparations : *une réparation est une requête qui ne génère plus de conflit et qui a une généralisation commune avec la requête initiale.*

Mots Clef

Coopération Personne-Système, Représentation de connaissances, Diagnostic, Explications.

Abstract

A mediator is a knowledge-based system which combines, in a centralized and homogeneous way, knowledge on an application domain and the description of the contents of different data sources relevant and available for this domain. In this paper, we work on queries posed to a mediator and which get no answer. First, we characterize as conflicts the minimal causes for the absence of answer to a query. Then, we produce its repairs' *set: a* repair *is a query which does not generate any* conflict *and which has a common generalization with the initial query.*

Keywords

Human-Machine Cooperation, Knowledge representation, Diagnosis, Explanations.

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 hétérogènes, réparties et autonomes, qui peuvent contenir des informations pertinentes pour une application. Par exemple, dans un domaine relatif au tourisme, on peut disposer d'une source d'information contenant les restaurants gastronomiques situés à Paris, d'une autre source fournissant la liste des monuments historiques et de leurs photos, et enfin d'une troisième source permettant d'accéder au plan de la ville de Paris. Obtenir des réponses à une requête aussi simple que la recherche d'un restaurant gastronomique avec vue sur la Tour Eiffel nécessite de combiner des éléments de réponses provenant des différentes sources. Décider quelles sources combiner et comment les combiner pour satisfaire une requête devient alors un problème de raisonnement sur des requêtes, sur des connaissances décrivant un domaine d'application (e.g. le tourisme), et sur des connaissances décrivant le contenu de plusieurs sources d'information.

Récemment, de nombreux travaux se sont intéressés au problème de l'intégration de sources d'informations, et des systèmes tels que Information Manifold [7], Sims [1] ou PICSEL [1] [6] ont vu le jour. L'approche commune suivie par les concepteurs de ces systèmes consiste à voir l'intégration d'informations comme la construction de *médiateurs* entre des utilisateurs et différents services d'informations relatifs à un même domaine d'application. Un médiateur est un système à base de connaissances qui regroupe, dans un cadre unificateur et homogène, des connaissances propres au domaine d'application et la description sémantique et abstraite du contenu des différentes sources d'informations disponibles. Les données restent stockées et réparties au niveau des sources d'informations sous une forme qui peut varier d'une source à l'autre. Le médiateur joue le rôle d'interface entre l'utilisateur et les sources en lui donnant l'impression qu'il interroge un système d'information centralisé et homogène. Sa tâche est de reformuler la requête posée par l'utilisateur dans le vocabulaire du domaine, en un ensemble de requêtes spécialisées exprimées dans le vocabulaire des différentes sources d'information disponibles.

Dans cet article, nous nous intéresserons au problème de la réparation des requêtes qui n'obtiennent pas de

1. PICSEL est financé par le CNET(Centre National d'Études des Télécommunications) sous le contrat 97 1B 378.

réponse. Ce cas arrive fréquemment dans un contexte d'intégration de sources de données où l'utilisateur, ne connaissant pas les limites de compétences des différentes sources, peut poser des requêtes qui, tout en ayant un sens dans le domaine d'application, ne peuvent être satisfaites par les sources de données disponibles. Il s'agit alors de modifier la requête initiale en la remplaçant par une requête moins contrainte pour qu'elle puisse obtenir des réponses, mais qui en reste sémantiquement proche pour que ses réponses satisfassent l'utilisateur.
Notre travail s'inscrit dans une approche fondée sur la logique qui présente l'intérêt de fournir une sémantique formelle claire et rigoureuse à la description du domaine et du contenu des sources, associée à des algorithmes d'inférence permettant de tester la satisfiabilité d'une requête et de calculer l'ensemble des réponses à une requête donnée. Dans ce cadre, le problème de la réparation de requêtes peut être vu comme un problème de généralisations de formules logiques.
La contribution de ce travail est double. D'une part, nous caractérisons les causes minimales de l'absence de réponses à une requête en termes de *conflits* regroupant les règles et contraintes responsables de l'incompatibilité de la requête avec la théorie du domaine et la description des sources. D'autre part, nous construisons un ensemble de *réparations :* une réparation est une requête qui ne conduit pas à des conflits et qui a une généralisation commune avec la requête initiale.

2 Préliminaires et exemples

Nous précisons tout d'abord le cadre logique dans lequel nous nous situons, en décrivant le formalisme utilisé pour exprimer les connaissances prises en compte sur le domaine d'application, le contenu des sources relatives à ce domaine, ainsi que les requêtes.

2.1 Connaissances du domaine

L'expression des connaissances du domaine s'appuie sur une représentation déclarative de classes d'objets (Voyage, Séjour, Vol, Pays...) et de relations entre ces classes. Les connaissances du domaine s'expriment à partir d'atomes de la forme $p(\bar{X})$ où p est un nom de relation et $\bar{X}$ un n-uplet de variables ou de constantes. Certaines relations unaires, qu'on appellera *concepts*, sont distinguées car elles représentent les classes d'objets pertinentes dans le domaine d'application. $C(x)$ est un atome-concept si C est un concept.

Les connaissances du domaine sont réparties en trois ensembles :

- **Un ensemble de règles $\mathcal{D}$**, de la forme :

$$r \ : p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$$

où les $p_i(\bar{X}_i)$ et $q(\bar{Y})$ sont des atomes et où toutes les variables de $\bar{Y}$ apparaissent dans $\bar{X}_1 \cup \ldots \cup \bar{X}_n$.
Une règle permet de définir une relation q (d'arité quelconque) en fonction d'autres relations.
ex : Le pays de départ d'un vol est celui de sa ville de départ. $\mathrm{Vol(x)} \wedge \mathrm{VilleDép(x,y)} \wedge \mathrm{VilleDe(p,y)} \Rightarrow \mathrm{PaysD(x,p)}$

Parmi les règles, on distingue celles qui traduisent une hiérarchie entre les concepts du domaine, de la forme : $C_1(x) \Rightarrow C_2(x)$, où C_1 et C_2 sont des concepts.
ex : Un vol direct est un vol. $\mathrm{VolDirect(X)} \Rightarrow \mathrm{Vol(X)}$

Les concepts d'une part, et les relations n'apparaissant pas en conclusion de règle d'autre part, sont dits *relations de base*. Les autres relations sont dites *ordinaires*.

- **Un ensemble de faits $\mathcal{F}$**, de la forme $p(\bar{a})$, où $\bar{a}$ est un n-uplet de constantes.

ex : VilleDe(Turquie, Ankara), VilleDe(Chypre, Nicosie)

- **Un ensemble de contraintes $\mathcal{C}$**, de la forme

$$c \ : l_1(\bar{X}_1) \wedge \ldots \wedge l_n(\bar{X}_n) \Rightarrow \perp$$

où $l_1(\bar{X}_1), \ldots, l_n(\bar{X}_n)$ sont des **littéraux** (des atomes ou la négation d'atomes).
Ces contraintes se répartissent en trois sous-ensembles.

- **un ensemble $\mathcal{C}_e$ de contraintes d'exclusion** de la forme $C_1(x) \wedge C_2(x) \Rightarrow \perp$. Elles permettent d'exprimer des exclusions entre concepts du domaine.
ex : Un voyage n'est pas un séjour. $\mathrm{Voyage(x)} \wedge \mathrm{Séjour(x)} \Rightarrow \perp$

- **un ensemble $\mathcal{C}_t$ de contraintes de typage** de la forme $p(\bar{X}) \wedge \neg C(x_i) \Rightarrow \perp$ avec $x_i \in \bar{X}$. Elles permettent de préciser à quel concept C est lié chaque argument de $p(\bar{X})$.
ex : La relation binaire VilleDépart prend en premier argument un voyage et en second argument un lieu géographique.
$\mathrm{VilleDépart(x,y)} \wedge \neg \mathrm{Voyage(x)} \Rightarrow \perp$
$\mathrm{VilleDépart(x,y)} \wedge \neg \mathrm{LieuGéographique(y)} \Rightarrow \perp$

- **un ensemble $\mathcal{C}_s$ de contraintes sémantiques,** qui regroupent toutes les contraintes ne rentrant pas dans les deux catégories précédentes, c-à-d celles qui expriment des contraintes générales sur le domaine.
ex : Il n'existe pas de vol direct entre deux pays en conflit. $\mathrm{VolDirect(x)} \wedge \mathrm{PaysD(x,}p_1) \wedge \mathrm{PaysA(x,}p_2)$
$\wedge \mathrm{EnConflit}(p_1, p_2) \Rightarrow \perp$

La sémantique de ces différents ensembles est la sémantique standard de la logique du premier ordre. Une interprétation $\mathcal{I}$ est définie sur un domaine non vide $\mathcal{O}^{\mathcal{I}}$. Pour chaque constante a, on assigne un objet $a^{\mathcal{I}} \in \mathcal{O}^{\mathcal{I}}$, et pour chaque relation d'arité n, on assigne une relation d'arité n sur le domaine $\mathcal{O}^{\mathcal{I}}$.

- $\mathcal{I}$ est un modèle d'une règle r si pour chaque assignation α des variables de la règle r à des éléments du domaine $\mathcal{O}^{\mathcal{I}}$, telle que $\alpha(\bar{X}_i) \in {p_i}^{\mathcal{I}}$ pour chaque p_i, on a $\alpha(\bar{Y}) \in q^{\mathcal{I}}$.

- Un fait $p(\bar{a})$ est vrai dans $\mathcal{I}$ ssi $\bar{a}^{\mathcal{I}} \in p^{\mathcal{I}}$.

- $\mathcal{I}$ est un modèle d'une contrainte c si pour chaque assignation α des variables de c avec des éléments du domaine $\mathcal{O}^{\mathcal{I}}$, on n'a pas simultanément pour tout l_i: $\alpha(\bar{X}_i) \in {l_i}^{\mathcal{I}}$.

- $\mathcal{I}$ est un modèle de $\mathcal{D}, \mathcal{F}$ et $\mathcal{C}$ s'il est un modèle de chacun de ses éléments.

2.2 Description des Sources

Le contenu de chaque source $\mathcal{S}$ est représenté à partir d'un vocabulaire $\mathcal{V}_S$ constitué d'autant de noms de relations locales v_i que de relations de base du domaine dont on sait que la source $\mathcal{S}$ fournit des instances. Ces relations locales v_i sont appelées des **vues**. La description du contenu d'une source $\mathcal{S}$ en termes de ses vues est constituée de :

- **un ensemble $\mathcal{D}_v$ d'implications logiques,** reliant chaque vue à la relation du domaine dont elle peut fournir des instances, $v_i(\bar{X}) \Rightarrow p(\bar{X})$.

ex : Les implications suivantes expriment le fait que la source $\mathcal{S}_1$ fournit des instances du concept *Hôtel* ainsi que des instances de la relation binaire *Localisé*.

$\mathcal{V}_{S_1}$: V_{11}, V_{12},
$V_{11}(x) \Rightarrow$ Hôtel(x)
$V_{12}(x,y) \Rightarrow$ Localisé(x, y)

- **un ensemble $\mathcal{C}_v$ de contraintes** caractérisant les instances des vues : $l_1(\bar{X}_1) \wedge \ldots \wedge l_n(\bar{X}_n) \Rightarrow \perp$, où les l_i sont des noms de relations de base et/ou des vues ou leur négation.

ex : Pour préciser que les localisations issues de $\mathcal{S}_1$ ne concernent que les hôtels présents dans $\mathcal{S}_1$, et que ceux-ci sont tous localisés aux Caraïbes, on écrira :

$V_{12}(x,y) \wedge \neg V_{11}(x) \Rightarrow \perp$
$V_{11}(x) \wedge Localis\acute{e}(x,y) \wedge \neg AuxCara\ddot{\imath}bes(y) \Rightarrow \perp$

La sémantique formelle des éléments de $\mathcal{D}_v$ et $\mathcal{C}_v$ est la même que celle décrite pour les connaissances du domaine.

2.3 Requêtes et réécritures de requêtes

Les requêtes que nous considérons sont des unions de requêtes conjonctives, c-à-d des unions de conjonctions d'atomes de la forme :

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

où les p_i sont des noms de relations du domaine apparaissant dans $\mathcal{D} \cup \mathcal{C}$. Les variables $\bar{X} = \bar{X}_1 \cup \ldots \cup \bar{X}_n$ sont appelées *variables distinguées* de la requête et représentent les variables dont l'utilisateur veut obtenir des instances quand il pose la requête. Les variables non distinguées $\bar{Y} = \bar{Y}_1 \cup \ldots \cup \bar{Y}_n$ servent à contraindre les variables distinguées ; $\bar{a}_1, \ldots, \bar{a}_n$ sont des n-uplets de constantes. Les variables sont quantifiées existentiellement.

Classiquement, poser une requête $Q(\bar{X})$ se définit relativement à une base de données $\mathcal{BD}$, éventuellement associée à une théorie du domaine $\mathcal{D}$, et consiste à déterminer si :

$$\mathcal{BD}, \mathcal{D} \models \exists \bar{X},\ \exists \bar{Y}\ [p_1(\bar{X}_1, \bar{Y}_1, \bar{a}_1) \wedge \ldots \wedge p_n(\bar{X}_n, \bar{Y}_n, \bar{a}_n)]$$

Dans le cas d'une réponse positive à cette question logique, la *réponse* à la requête est l'ensemble des n-uplets $\bar{b} = \bar{b}_1 \cup \ldots \cup \bar{b}_n$ tels que :

$$\mathcal{BD}, \mathcal{D} \models \exists \bar{Y}\ [p_1(\bar{b}_1, \bar{Y}_1, \bar{a}_1) \wedge \ldots \wedge p_n(\bar{b}_n, \bar{Y}_n, \bar{a}_n)]$$

Or, nous ne sommes pas dans la situation classique : dans une approche de type médiateur, nous n'avons pas accès aux contenus des sources de données mais à une abstraction de celles-ci, exprimée sous la forme d'un ensemble $\mathcal{V}$ de vues. Poser une requête à un médiateur consiste alors à rechercher les différentes réécritures de cette requête en termes de vues. Une réécriture d'une requête $Q(\bar{X})$ est une requête qui, avec la théorie du domaine, implique logiquement $Q(\bar{X})$.

Définition 2.1: *Soit $Q(\bar{X})$ une requête. Soit $\mathcal{D} \cup \mathcal{C}$ la description du domaine et $\mathcal{D}_{\mathcal{V}} \cup \mathcal{C}_{\mathcal{V}}$ la description des sources relatives à ce domaine. Soit $Q_{\mathcal{R}}$ une requête conjonctive.*

- *$Q_{\mathcal{R}}(\bar{X})$ est une réécriture de $Q(\bar{X})$ ssi*
  $Q_{\mathcal{R}}(\bar{X}), \mathcal{D} \cup \mathcal{C} \cup \mathcal{D}_{\mathcal{V}} \cup \mathcal{C}_{\mathcal{V}}$ est satisfiable, et
  $Q_{\mathcal{R}}(\bar{X}), \mathcal{D} \cup \mathcal{C} \cup \mathcal{D}_{\mathcal{V}} \cup \mathcal{C}_{\mathcal{V}} \models Q(\bar{X})$

- *$Q_{\mathcal{R}}(\bar{X})$ est une réécriture terminale de $Q(\bar{X})$ ssi $Q_{\mathcal{R}}(\bar{X})$ est une réécriture de $Q(\bar{X})$ qui ne contient que des atomes-vues.*

L'ensemble des réécritures terminales d'une requête fournit les plans de requêtes spécialisés dont l'exécution permet d'obtenir l'ensemble des réponses de la requête initiale.

3 Position du problème

Dans le contexte du médiateur Picsel [6], où nous nous situons, un algorithme de calcul des réécritures terminales a été conçu et mis en œuvre en Java. Ce calcul procède par expansions successives de la requête initiale. Un pas d'expansion consiste en un pas de chaînage-arrière sur les règles décrivant le domaine et si possible les implications déterminant le lien entre les vues et les relations du domaine. Chaque pas d'expansion fournit des réécritures potentielles de la requête initiale. Le calcul s'arrête quand on n'obtient que des réécritures terminales ou des réécritures qui n'ont pas d'expansion possible. Pour chaque réécriture potentielle obtenue à chaque pas d'expansion, on teste sa compatibilité avec les contraintes $\mathcal{C}$ et $\mathcal{C}_v$.

Nous intervenons quand le médiateur détermine que toutes les réécritures potentielles calculées sont insatisfiables avec les contraintes du domaine et/ou sur les vues. A cet instant, sans avoir accès aux sources de données, on est sûr que la requête posée par l'utilisateur n'a pas de réponses. Les connaissances du domaine et la description des sources d'information sont supposées consistantes, c'est donc la requête qui est responsable de l'insatisfiabilité.
Plus précisement, nos algorithmes prennent comme entrée des requêtes conjonctives $Q(\bar{X})$ telles que :

$$Q(\bar{X}), \mathcal{D} \cup \mathcal{D}_{\mathcal{V}} \cup \mathcal{C} \cup \mathcal{C}_{\mathcal{V}} \models \perp \quad et \quad \mathcal{D} \cup \mathcal{D}_{\mathcal{V}} \cup \mathcal{C} \cup \mathcal{C}_{\mathcal{V}} \not\models \perp.$$

Chaque $Q(\bar{X})$ considérée est une réécriture potentielle de la requête initiale de l'utilisateur, calculée par le médiateur PICSEL et repérée par lui comme insatisfiable. Il est important de noter que $Q(\bar{X})$ peut très bien être simplement la requête initiale, détectée comme étant insatisfiable avant tout pas d'expansion.

Exemple : Supposons que l'on dispose de deux sources, l'une sur des séjours à plus de 4000 francs, l'autre sur des voyages à plus de 8000 francs,

S_1 $V_{11}(x) \Rightarrow Séjour(x)$
$V_{11}(x) \wedge Prix(x,y) \wedge \neg Plus4000F(y) \Rightarrow \perp$
S_2 $V_{21}(x) \Rightarrow Voyage(x)$
$V_{21}(x) \wedge Prix(x,y) \wedge \neg Plus8000F(y) \Rightarrow \perp$

de deux règles disant que les séjours et les voyages sont des produits,

r_1 : $Séjour(x) \Rightarrow Produit(x)$
r_2 : $Voyage(x) \Rightarrow Produit(x)$

et des contraintes d'exclusion suivantes :

c_1 : $Voyage(x) \wedge Séjour(x) \Rightarrow \perp$
c_2 : $Plus4000F(x) \wedge Moins3000F(x) \Rightarrow \perp$
c_2 : $Plus8000F(x) \wedge Moins3000F(x) \Rightarrow \perp$

Dans sa requête, l'utilisateur souhaite obtenir *un séjour qui soit également un voyage* ou *un produit à moins de 3000Fr.*

$Q(x)$: ($Séjour(x) \wedge Voyage(x)$)
$\vee$ ($Produit(x) \wedge Prix(x,y) \wedge Moins3000F(y)$).

Le médiateur nous transmet les réécritures potentielles suivantes qui sont toutes insatisfiables avec les règles $\mathcal{D}$ et $\mathcal{D}_v$ et les contraintes $\mathcal{C}$ et $\mathcal{C}_v$:

$Q_{\mathcal{R}_1}$: $Séjour(x) \wedge Voyage(x)$,
$Q_{\mathcal{R}_2}$: $V_{11}(x) \wedge Prix(x,y) \wedge Moins3000F(y)$,
$Q_{\mathcal{R}_3}$: $V_{21}(x) \wedge Prix(x,y) \wedge Moins3000F(y)$.

Notre tâche est d'identifier les causes d'insatisfiabilité des requêtes qui nous sont fournies, et de proposer des corrections qui les rendent satisfiables. Identifier ces causes c'est déterminer pourquoi ces requêtes sont incompatibles avec les contraintes, c-à-d trouver pour chacune d'elles, la ou les contraintes violées. Pour ce faire, on sature pas à pas les requêtes fournies par PICSEL avec les règles et les contraintes du domaine jusqu'à identifier l'étape où apparaît l'inconsistance, traduite par des conflits (section 4). Le calcul des réparations adaptées à ces conflits est décrit dans la section 5.

4 Détection des conflits

Cette section présente tout d'abord l'opérateur utilisé pour saturer des requêtes conjonctives, puis la définition d'un conflit et enfin la façon dont il est détecté.

4.1 Opérateur de saturation de requête

L'opérateur de saturation de requête est défini à partir de deux opérateurs de saturation sur un ensemble de littéraux $\mathcal{L}$. Le premier opérateur sature le sous-ensemble $\mathcal{A}$ des littéraux positifs de $\mathcal{L}$ à partir de l'ensemble de règles. Le second opérateur sature l'ensemble de littéraux $\mathcal{L}$ à partir des différentes contraintes du domaine.

Définition 4.1: *Étant donné un ensemble $\mathcal{D} \cup \mathcal{D}_v$ de règles, et un ensemble $\mathcal{A}$ d'atomes, on définit l'opérateur d'inférence immédiate T de la façon suivante :*

$$T(\mathcal{A}) = \bigcup_{r \in \mathcal{D} \cup \mathcal{D}_v} T_r(\mathcal{A})$$

où $T_r(\mathcal{A}) = \mathcal{A} \cup q$, si les atomes en condition de r sont unifiables avec des atomes de $\mathcal{A}$ (q étant la conclusion de la règle unifiée),
et $T_r(\mathcal{A}) = \mathcal{A}$ sinon.

Définition 4.2: *Étant donné un ensemble $\mathcal{C} \cup \mathcal{C}_v$ de contraintes d'intégrité de la forme $c : \wedge_{i=1..n} li \Rightarrow \perp$, et un ensemble de littéraux $\mathcal{L}$, on définit l'opérateur $T_C(\mathcal{L})$ d'ajout de résidu de la façon suivante :*

$$T_C(\mathcal{L}) = \mathcal{L} \cup \bigcup_{c \in \mathcal{C} \cup \mathcal{C}_v} r_c(\mathcal{L}),$$

où $r_c(\mathcal{L}) = \emptyset$ si au moins deux littéraux de c ne sont pas unifiables avec des littéraux de $\mathcal{L}$.
et $r_c(\mathcal{L}) = \bigcup_\sigma \{not(l_j)\sigma$ si (n-1) littéraux l_i, $i \neq j$, sont unifiables par σ (plus grand unificateur p.g.u.) avec des littéraux de $\mathcal{L}\}$.

Exemple : $c = ours(x) \wedge blanc(x) \wedge \neg polaire(x) \Rightarrow \perp$,
avec $\mathcal{L}_1 = \{ours(x)\}$, on obtient $r_c(\mathcal{L}_1) = \emptyset$.
avec $\mathcal{L}_2 = \{ours(x), \neg polaire(x)\}$, $r_c(\mathcal{L}_2) = \{\neg blanc(x)\}$.
avec $\mathcal{L}_3 = \{ours(x), blanc(x), \neg polaire(x)\}$,
$r_c(\mathcal{L}_3) = \{\neg ours(x), \neg blanc(x), polaire(x)\}$.

À partir de ces deux opérateurs de saturation, on peut maintenant définir l'opérateur de saturation de requête.

Définition 4.3: *L'opérateur T_S de saturation de la requête est défini par récurrence de la façon suivante :*

- $T_S(\mathcal{L}) = T(T_C(\mathcal{L}))$
- $T_{S,1}(\mathcal{L}) = T_S(\mathcal{L})$
- $T_{S,n+1}(\mathcal{L}) = T_S(T_{S,n}(\mathcal{L}))$

Proposition 4.1:
Cette famille d'opérateurs a un plus petit point fixe : il existe n tel que $\forall m \geq n$, $T_{S,m}(\mathcal{L}) = T_{S,n}(\mathcal{L})$.
$T_{S,n}(\mathcal{L})$ *est appelé le saturé de* $\mathcal{L}$ *par* $\mathcal{D} \cup \mathcal{D}_v \cup \mathcal{C} \cup \mathcal{C}_v$.

La proposition suivante caractérise l'insatisfiabilité d'une requête.

Proposition 4.2: *Une requête* $Q(\bar{X})$ *est insatisfiable ssi il existe un atome* a_i *tel que* a_i *et* $\neg a_i$ *appartiennent au saturé de* $Q(\bar{X})$ *par* $\mathcal{D} \cup \mathcal{D}_v \cup \mathcal{C} \cup \mathcal{C}_v$.

Remarquons que la requête initiale ne contenant que des atomes et l'opérateur d'inférence immédiate T n'ajoutant que des atomes au saturé, l'introduction de littéraux négatifs ne peut venir que de l'utilisation de l'opérateur T_C d'ajout de résidu sur au moins une contrainte. L'existence d'une inconsistance implique donc le viol d'au moins une contrainte.

4.2 Définition d'un conflit

On note Q_1 un sous-ensemble d'atomes d'une requête conjonctive $Q(\bar{X})$, $\mathcal{D}_1$ un sous-ensemble des connaissances $\mathcal{D} \cup \mathcal{D}_v$, et $\mathcal{C}_1$ un sous-ensemble des contraintes $\mathcal{C} \cup \mathcal{C}_v$.

Définition 4.4: *Un conflit est un triplet* $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *tel que* $Q_1, \mathcal{D}_1, \mathcal{C}_1 \models \perp$ *, où* $(\mathcal{D}_1, \mathcal{C}_1)$ *est appelé cause du conflit.*

On distingue les conflits entre eux selon deux critères. Le premier critère repose sur l'ensemble des atomes de la requête apparaissant dans le conflit.

Définition 4.5: *Un conflit* $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *est Q_minimal si pour tout* $Q_2 \subset Q_1$ *(strictement) il n'existe pas* $(\mathcal{D}_2, \mathcal{C}_2)$ *tel que* $Q_2, \mathcal{D}_2, \mathcal{C}_2 \models \perp$

Exemple

$\mathcal{C}$	c_1 : a(x) $\wedge$ b(x) $\wedge$ c(x) $\Rightarrow \perp$	c_2 : f(x) $\wedge$ g(x) $\Rightarrow \perp$
$\mathcal{D}$	r_1 : a(x) $\Rightarrow$ f(x)	r_2 : b(x) $\Rightarrow$ g(x)

Q(x) :	a(x) $\wedge$ b(x) $\wedge$ c(x)
$Conflit_1$:	({a(x) , b(x) , c(x) },{},{c_1})
$Conflit_2$:	({a(x) , b(x) },{r_1,r_2},{c_2})
$Q_2 \subset Q_1$	$Conflit_1$ n'est pas Q_minimal

Le second critère porte sur les ensembles de connaissances apparaissant dans le conflit. On définit pour cela un opérateur d'inclusion stricte sur les causes de conflit.

Définition 4.6 : *L'opérateur d'inclusion stricte sur les causes est défini de la façon suivante :*
$(\mathcal{D}_1, \mathcal{C}_1) \subset (\mathcal{D}_2, \mathcal{C}_2)$ *ssi*
$(\mathcal{D}_1 \subset \mathcal{D}_2$ *et* $\mathcal{C}_1 \subseteq \mathcal{C}_2)$ *ou* $(\mathcal{D}_1 \subseteq \mathcal{D}_2$ *et* $\mathcal{C}_1 \subset \mathcal{C}_2)$.

Définition 4.7: *Une cause minimale de conflit pour un ensemble d'atomes* Q_1 *est un couple* $(\mathcal{D}_1, \mathcal{C}_1)$ *tel qu'il n'existe pas de cause* $(\mathcal{D}_2, \mathcal{C}_2) \subset (\mathcal{D}_1, \mathcal{C}_1)$ *telle que* $Q_1, \mathcal{D}_2, \mathcal{C}_2 \models \perp$.

A partir des définitions 4.5 et 4.7, on distingue parmi tous les conflits détectables ceux qui sont pertinents.

Définition 4.8: *Un conflit* $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *est pertinent s'il est Q_minimal et si* $(\mathcal{D}_1, \mathcal{C}_1)$ *est une cause minimale de conflit pour* Q_1.

Exemple

$\mathcal{D}$:	r_1 père(x) $\Rightarrow$ homme(x)	r_2 mère(x) $\Rightarrow$ femme(x)
	r_3 père(x) $\Rightarrow$ adulte(x)	r_4 mère(x) $\Rightarrow$ adulte(x)
$\mathcal{C}$:	c_1 homme(x) $\wedge$ femme(x) $\Rightarrow \perp$	
	c_2 adulte(x) $\wedge$ enfant(x) $\Rightarrow \perp$	
Q(x) : père(x), mère(x), enfant(x)		

Conflits pertinents	({père(x), mère(x)},{r_1, r_2},{c_1})
	({père(x), enfant(x)},{r_3},{c_2})
	({mère(x), enfant(x)},{r_4},{c_2})

4.3 Calcul des conflits

Les opérateurs définis précédemment permettent de saturer une requête conjonctive à partir des contraintes et des règles du domaine. Un conflit est détecté si la saturation de la requête conduit à ajouter au saturé un littéral l_i tel que sa négation $\neg l_i$ soit déjà présente.

Identifier un conflit consiste à calculer l'origine des littéraux en conflit, c-à-d à retrouver les atomes de la requête initiale (appelés atomes initiaux) dont ont été dérivés les deux littéraux considérés, ainsi que les connaissances utilisées pour ces dérivations.

Pour ce faire, chaque nouveau littéral ajouté dans une étape de saturation est indicé par la connaissance (une contrainte c ou une règle r) qui a permis sa déduction. Un atome initial est indicé par lui-même.
- $indice(l_i) = c$, si l_i a été obtenu par utilisation de l'opérateur de calcul de résidu r_c.
- $indice(l_i) = r$, si l_i a été obtenu par utilisation de l'opérateur d'inférence immédiate T_r.
- $indice(l_i) = l_i$, si l_i est un atome initial.

Si un littéral peut être inféré à partir de plusieurs contraintes ou règles, on considèrera, pour simplifier la présentation, qu'on a autant de littéraux distincts que de contraintes ou règles ayant permis sa déduction.

L'**origine** d'un littéral l_i, $org(l_i)$, est un triplet $(Q_i, \mathcal{D}_i, \mathcal{C}_i)$ où Q_i est un ensemble d'atomes initiaux et $Q_i, \mathcal{D}_i, \mathcal{C}_i \models l_i$.

On définit l'opérateur $\oplus$ sur deux origines, tel que $(Q_i, \mathcal{D}_i, \mathcal{C}_i) \oplus (Q_j, \mathcal{D}_j, \mathcal{C}_j) = (Q_i \cup Q_j, \mathcal{D}_i \cup \mathcal{D}_j, \mathcal{C}_i \cup \mathcal{C}_j)$

Une origine se calcule de la façon suivante :

- $org(l_i) = (\{l_i\}, \{\}, \{\})$, si indice($l_i$)=$l_i$.
- $org(l_i) = \bigoplus_{j=1...n} org(l_j) \oplus (\{\}, \{r\}, \{\})$, si $indice(l_i) = r$ et $r = \wedge_{j=1..n} l_j \Rightarrow l_i$.

– $org(l_i) = \bigoplus_{j=1..n, i \neq j} org(l_j) \oplus (\{\},\{\},\{c\})$, si $indice(l_i) = c$ et $c = \bigwedge_{j=1..n} l_j \Rightarrow \bot$.

Exemple :

$\mathcal{C}$	c_1 : f(x) $\wedge$ g(x) $\Rightarrow \bot$
$\mathcal{D}$	r_1 : a(x) $\Rightarrow$ f(x) r_2 : b(x) $\Rightarrow$ g(x)
	r_3 : c(x) $\Rightarrow$ f(x)
Q(x)	a(x) $\wedge$ b(x) $\wedge$ c(x)
saturé	$\{a(x), b(x), c(x), f(x), g(x), \neg g(x), \neg f(x)\}$

$indice(f(x)_1) : r_1$	$org(f(x)_1)$:	$(\{a(x)\},\{r_1\},\{\})$
$indice(f(x)_2) : r_3$	$org(f(x)_2)$:	$(\{c(x)\},\{r_3\},\{\})$
$indice(g(x)) : r_2$	$org(g(x))$:	$(\{b(x)\},\{r_2\},\{\})$
$indice(\neg g(x)_1) : c_1$	$org(\neg g(x)_1)$:	$(\{a(x)\},\{r_1\},\{c_1\})$
$indice(\neg g(x)_2) : c_1$	$org(\neg g(x)_2)$:	$(\{c(x)\},\{r_3\},\{c_1\})$
$indice(\neg f(x)) : c_1$	$org(\neg f(x))$:	$(\{b(x)\},\{r_2\},\{c_1\})$

Un conflit dû à la présence conjointe dans le saturé courant des littéraux l_i et $\neg l_i$ est calculé de la façon suivante : $Conflit_{i,\neg i} = org(l_i) \oplus org(\neg l_i)$.

Sur l'exemple précédent, on trouve les 4 conflits suivants :

$$\begin{aligned}
Conflit_{g,\neg g_1} &= org(g(x)) \oplus org(\neg g(x)_1) \\
&= (\{a(x), b(x)\},\{r_1, r_2\},\{c_1\}) \\
Conflit_{g,\neg g_2} &= (\{c(x), b(x)\},\{r_3, r_2\},\{c_1\}) \\
Conflit_{f_1,\neg f} &= (\{a(x), b(x)\},\{r_1, r_2\},\{c_1\}) \\
Conflit_{f_2,\neg f} &= (\{c(x), b(x)\},\{r_3, r_2\},\{c_1\})
\end{aligned}$$

4.4 Sélection des conflits pertinents

Lors de la détection et du calcul d'un nouveau conflit, $Conflit_n : (Q_n, \mathcal{D}_n, \mathcal{C}_n)$, celui-ci doit être comparé aux éventuels autres conflits $(Q_i, \mathcal{D}_i, \mathcal{C}_i)$ précédemment identifiés.

- S'il existe $Conflit_i$ tel que $Q_i \subset Q_n$, $Conflit_n$ n'est pas $Q_{minimal}$ et doit être ignoré.
- Sinon, s'il existe $Conflit_i$ tel que $Q_i = Q_n$ et si $\mathcal{D}_i \subseteq \mathcal{D}_n$ et $\mathcal{C}_i \subseteq \mathcal{C}_n$ la cause de $Conflit_n$ n'est pas minimale, ou déjà connue, et ce conflit est ignoré,
- Sinon, pour chaque $Conflit_i$, si $Q_i = Q_n$ et $(\mathcal{D}_n, \mathcal{C}_n) \subset (\mathcal{D}_i, \mathcal{C}_i)$, ou si $Q_i \supset Q_n$, $Conflit_i$ n'est plus pertinent et est abandonné.

Sur l'exemple précédent, ne seront gardés que les conflits $Conflit_{g,\neg g_1}$ et $Conflit_{g,\neg g_2}$.

5 Réparation des conflits

Considérons une requête $Q(\bar{X})$ qui présente des conflits. Il s'agit de déterminer un ensemble de réparations qui les suppriment. Une réparation de $Q(\bar{X})$ est une requête $Q'(\bar{X})$ pour laquelle il n'existe pas de conflit et telle que $Q(\bar{X})$ et $Q'(\bar{X})$ ont une généralisation commune $Q''(\bar{X})$. La généralisation que nous proposons est fondée sur la notion de généralisant de concept et nécessite un certain nombre de définitions intermédiaires pour aboutir à la notion de *généralisation valide* d'un atome-concept pour une contrainte.

5.1 Généralisation valide

Définition 5.1: *Un généralisant direct d'un concept $\mathcal{C}$ est un concept $\mathcal{C}'$ tel que la règle $\mathcal{C}(x) \Rightarrow \mathcal{C}'(x)$ appartient à $\mathcal{D}$.*

Un concept peut avoir plusieurs généralisants directs.

Définition 5.2: *$\mathcal{C}'$ 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 $\mathcal{C}''$ un généralisant de $\mathcal{C}$.*

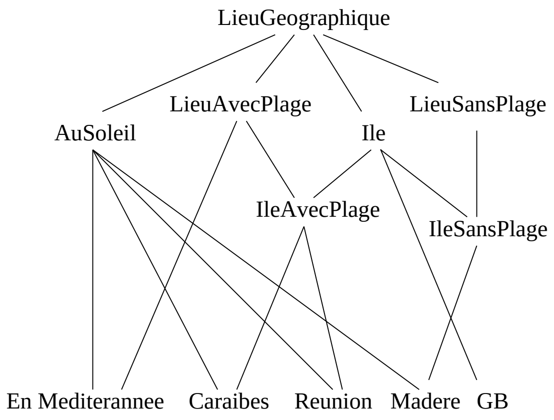

FIG. 1 – *Hiérarchie des lieux géographiques*

Dans la figure 1 qui présente une partie de la hiérarchie de concepts exprimée dans les règles du domaine, les généralisants du concept *Réunion* sont *AuSoleil* et *ÎleAvecPlage* pour ses généralisants directs et *Île*, *LieuAvecPlage* et *LieuGéographique*.

Les relations ordinaires n'apparaissent pas dans la hiérarchie de concepts et ne sont pas prises en compte par la notion de généralisant. On introduit donc la notion de **dépendant** d'une relation.

Définition 5.3: *La relation $\mathcal{R}'$ est un dépendant direct d'une relation $\mathcal{R}$ s'il existe dans $\mathcal{D}$ une règle où $\mathcal{R}$ apparaît en condition et $\mathcal{R}'$ en conclusion.*

Définition 5.4: *$\mathcal{R}'$ est un dépendant d'une relation $\mathcal{R}$ si $\mathcal{R}'$ est un dépendant direct de $\mathcal{R}$ ou s'il existe $\mathcal{R}''$ tel que $\mathcal{R}'$ est un dépendant direct de $\mathcal{R}''$ et $\mathcal{R}''$ est un dépendant de $\mathcal{R}$.*

Ici, on souhaite généraliser un concept parce qu'il intervient dans une contrainte violée par la requête (soit directement, soit par le biais de l'un de ses généralisants ou de ses dépendants) et entraîne un conflit.

Définition 5.5 : *Un concept est* **neutre** *pour une contrainte si ce concept, ses généralisants et ses dépendants n'apparaissent pas dans la contrainte.*

Exemple : pour la contrainte c_1 affirmant qu'une baignade n'est pas un loisir praticable dans un lieu sans plage, c_1 : $LieuSansPlage(x) \wedge LoisirPraticable(x, y) \wedge BelleBaignade(y) \Rightarrow \perp$, le concept *Réunion* est neutre pour cette contrainte alors que *Madère* ne l'est pas.

Définition 5.6: *$\mathcal{C}'$ est un plus petit généralisant neutre de $\mathcal{C}$ pour une contrainte c, si $\mathcal{C}'$ est un généralisant de $\mathcal{C}$, neutre pour une contrainte c et s'il n'existe pas $\mathcal{C}''$, généralisant de $\mathcal{C}$, neutre pour la contrainte c, tel que $\mathcal{C}'$ est un généralisant de $\mathcal{C}''$.*

Définition 5.7: *Soit $\mathcal{C}$ un concept et $\mathcal{C}_1$, $\mathcal{C}_2$,...,$\mathcal{C}_n$, ses plus petits généralisants neutres pour la contrainte c, la généralisation neutre de l'atome-concept $\mathcal{C}(x)$ pour la contrainte c, est la conjonction $\mathcal{C}_1(x) \wedge \mathcal{C}_2(x) \wedge ... \wedge \mathcal{C}_n(x)$.*

Exemple : pour la contrainte c_1, la généralisation neutre de $Madère(x)$ est la conjonction $\hat{I}le(x) \wedge AuSoleil(x)$. [2]

Pour garder la cohérence de la requête après la généralisation, si dans celle-ci l'atome-concept $C(x)$ partage une variable avec un autre atome, de la forme $R(\bar{X})$, il faut tester si la généralisation neutre de $C(x)$ vérifie la contrainte de typage associée à cet atome.

Définition 5.8: *Une conjonction d'atome $\mathcal{C}_1(x) \wedge \mathcal{C}_2(x) \wedge ... \wedge \mathcal{C}_n(x)$ vérifie une contrainte de typage c_t : $R(\bar{X}) \wedge \neg C'(x_i) \Rightarrow \perp$, avec $x_i \in \bar{X}$, ssi le concept $\mathcal{C}'$ est un des concepts $\mathcal{C}_1$, $\mathcal{C}_2$,...,$\mathcal{C}_n$ ou un de ses généralisants.*

Exemple : supposons que l'utilisateur cherche un hôtel à Madère, où il soit possible de faire de belles baignades, Q(x) = Hôtel(x) ∧ Localisé(x,y) ∧ Madère(y) ∧ LoisirAss(x,z) ∧ BelleBaignade(z).
Dans cette requête, l'atome $Madère(y)$ est lié à l'atome $Localisé(x, y)$ dont le deuxième argument doit être un LieuGéographique. *LieuGéographique* étant un généralisant de *Île*, la généralisation $\hat{I}le(y) \wedge AuSoleil(y)$ vérifie la contrainte de typage associée à la relation *Localisé*.

Définition 5.9: *La généralisation valide d'un atome-concept $\mathcal{C}(x)$ pour une contrainte c et une requête $Q(\bar{X})$, est une généralisation neutre de $\mathcal{C}(x)$ pour c, qui vérifie les contraintes de typage associées aux atomes liés à l'atome-concept $\mathcal{C}(x)$ dans $Q(\bar{X})$.*

Remarquons qu'il n'y a qu'une généralisation neutre pour un atome-concept et une contrainte. Si celle-ci n'est pas valide pour la requête, l'atome n'a pas de généralisation valide pour la contrainte considérée.

[2]. Le concept *LieuGéographique* fait lui aussi théoriquement partie des plus petits généralisants neutres de *Madère*, mais il n'est pas conservé en tant que tel car il est aussi un généralisant de *Île* et *AuSoleil*, qui appartiennent déjà à la généralisation.

5.2 Réparation d'un conflit par généralisation d'atome-concept

Dans cette partie, on montre comment utiliser les généralisations valides pour réparer un conflit dans une requête.

Définition 5.10 : *Soit $Q(\bar{X})$ une requête. Une réparation d'un conflit $(Q_i, \mathcal{D}_i, \mathcal{C}_i)$ est un ensemble Q'_i, construit à partir de Q_i en remplaçant l'un de ses concepts par sa généralisation valide pour une des contraintes de $\mathcal{C}_i$ et $Q(\bar{X})$.*

On peut montrer que la réparation proposée est une généralisation de l'ensemble d'atomes Q_i pour laquelle il n'existe pas de conflit.

Proposition 5.1 : *Soit un conflit $(Q_i, \mathcal{D}_i, \mathcal{C}_i)$ et Q'_i une de ses réparations, on a les propriétés suivantes :*
$\mathcal{D}, Q_i \models Q'_i$ et $(Q'_i, \mathcal{D}_i, \mathcal{C}_i)$ n'est plus un conflit.

Exemple : avec les connaissances suivantes,
r_1 : Hôtel(x) ⇒ LieuDeRésidence(x)
r_2 : BelleBaignade(x) ⇒ ActivitéLoisir(x)
r_3 : Madère(x) ⇒ ÎleSansPlage(x)
c_1 : LieuSansPlage(y) ∧ LoisirPraticable(y,z) ∧ BelleBaignade(z) ⇒ ⊥
c_2 : LieuDeRésidence(x) ∧ LoisirAss(x,z) ∧ Localisé(x,y) ∧ LieuGéographique(y) ∧ ¬ LoisirPraticable(y, z) ⇒ ⊥
c_3 : LoisirAss(x, z) ∧ ¬ LieuDeRésidence(x) ⇒ ⊥
c_4 : LoisirAss(x, z) ∧ ¬ ActivitéLoisir(z) ⇒ ⊥
c_5 : Localisé(x, y) ∧ ¬ LieuGéographique(y) ⇒ ⊥

et la requête précédente, Q(x) : Hôtel(x) ∧ Localisé(x,y) ∧ Madère(y) ∧ LoisirAss(x,z) ∧ BelleBaignade(z).

On identifie, dans la requête l'unique conflit :
({Localisé(x,y), Madère(y), LoisirAss(x,z), BelleBaignade(z)}, $\{r_1, r_3\}, \{c_1, c_2, c_3\}$).
En remplaçant dans Q_i l'atome-concept $Madère(y)$ par sa généralisation valide pour c_1, $\hat{I}le(y) \wedge AuSoleil(y)$, on peut construire une réparation Q'_i(x) du conflit : {Localisé(x,y), **Île(y)**, **AuSoleil(y)**, LoisirAss(x,z), BelleBaignade(z)}.

Si Q_i contient n concepts et $\mathcal{C}_i$ m contraintes, le conflit pourrait avoir n*m réparations potentielles, mais chercher une généralisation valide d'un atome-concept pour une contrainte, n'a de sens que si ce concept (ou l'un de ses généralisants) apparaît dans la contrainte.
Dans la pratique, en partant des contraintes en cause dans le conflit, on travaille sur tous les généralisants des concepts de Q_i apparaissant dans la contrainte :
- pour c_1, les concepts concernés sont Madère et BelleBaignade. Pour Madère(y), on retrouve la réparation Q'_i(x) précédente, et pour BelleBaignade(z), sa généralisation valide, ActivitéLoisir(z), permet de construire la réparation Q''_i(X) : {Localisé(x,y), Madère(y), LoisirAss(x,z), **ActivitéLoisir(z)** }

- sur c_2, les atomes-concepts concernés sont LieuDeRésidence(x) et LieuGéographique(y) pour lesquels on ne trouve pas de généralisation valide.
- c_3 n'est pas exploitable car elle ne possède pas de concept sous une forme positive. Il n'y a donc que deux réparations possibles de ce conflit.

5.3 Autres types de réparation de conflit

Les réparations par généralisation de concept ne sont pas suffisantes, en particulier quand la requête est mal formulée par rapport à l'organisation des connaissances du domaine: les contraintes intervenant dans le conflit ne sont souvent que des contraintes de typage ou d'exclusion et ce type de réparation est en échec. Dans ce cas, il faut aussi travailler sur les atomes binaires $R(x,y)$ en cause dans les contraintes de typage.

Pour une requête de la forme $C_1(x) \wedge R(x,y) \wedge C_2(y)$ qui entraîne le conflit $(\{R(x,y), C_1(x)\}, \{\}, \{c_1, c_2\})$ où c_1 est la contrainte de typage: $R(x,y) \wedge \neg C_3(x) \Rightarrow \perp$, et c_2 la contrainte d'exclusion: $C_1(x) \wedge C_3(x) \Rightarrow \perp$, on peut effectuer deux sortes de réparation (dites par typage) si on privilégie la relation binaire ou le concept :
- la première conserve l'atome R(x, y) et consiste à typer correctement le concept qui lui est associé en remplaçant simplement $C_1(x)$ par $C_3(x)$.
- la deuxième conserve les concepts C_1 et C_2 et consiste à remplacer la relation R par une chaîne d'atomes qui relie correctement $C_1(x)$ à $C_2(y)$. La recherche des remplaçants se fait en étudiant les relations binaires associées aux concepts considérés. On se limitera à l'étude de chaînes d'au plus deux relations.

Exemple: Dans notre représentation du domaine, la relation *VilleArrivée* relie un *LieuGéographique* à un *Voyage*, et un *Voyage* n'est pas un *Séjour*. Un *Séjour* est quant à lui relié à un *LieuDeRésidence* par la relation *RésidenceAssociée* et un *LieuDeRésidence* est localisé dans un *LieuGéographique*.

c_1 $VilleArrivée(x,y) \wedge \neg Voyage(x) \Rightarrow \perp$
c_2 $VilleArrivée(x,y) \wedge \neg LieuGéographique(y) \Rightarrow \perp$
c_3 $RésidenceAssociée(x,y) \wedge \neg Séjour(x) \Rightarrow \perp$
c_4 $RésidenceAssociée(x,y) \wedge \neg LieuDeRésidence(y) \Rightarrow \perp$
c_5 $Localisé(x,y) \wedge \neg LieuDeRésidence(x) \Rightarrow \perp$
c_6 $Localisé(x,y) \wedge \neg LieuGéographique(y) \Rightarrow \perp$
c_7 $Séjour(x) \wedge Voyage(x) \Rightarrow \perp$

Nous savons aussi que *Séjour* et *Voyage* sont des *Produit* et *LaRéunion*, un *LieuGéographique*.

r_1 : $Séjour(x) \Rightarrow Produit(x)$
r_2 : $Voyage(x) \Rightarrow Produit(x)$
r_3 : $LaRéunion(x) \Rightarrow LieuGéographique(x)$

Supposons qu'un utilisateur demande *un séjour dont la ville d'arrivée est située à La Réunion*,

$Q(x) : Séjour(x) \wedge VilleArrivée(x,y) \wedge Réunion(y)$.

Sa requête est donc incompatible avec l'organisation du domaine et le conflit identifié est le suivant :

({ VilleArrivée(x,y), Séjour(x) },{},{c_1, c_7}).

La réparation par généralisation de concept ne permet de travailler que sur le concept *Séjour* dont un généralisant est *Produit*, concept neutre pour c_7. Mais la relation *VilleArrivée* ne s'associe pas à un *Produit* et cette généralisation n'est pas valide.

Les réparations par typage donnent les résultats suivants :
- si on privilégie la relation *VilleArrivée*, on obtient la réparation suivante : $Q'_1(x)$: **Voyage(x)** $\wedge$ VilleArrivée(x, y) $\wedge$ Réunion(y).
- si on privilégie les concepts *Séjour* et *Réunion*, il faut remplacer l'atome VilleArrivée(x, y) par une chaîne reliant Séjour(x) à Réunion(y). On trouvera qu'un Séjour a une RésidenceAssociée qui est un *LieuDeRésidence*, lequel doit être localisé dans un *LieuGéographique* ! *Réunion* étant un *LieuGéographique*, la réparation suivante peut être proposée : $Q'_2(x)$: Séjour(x) $\wedge$ **RésidenceAssociée(x, z)** $\wedge$ **LieuDeRésidence(z)** $\wedge$ **Localisé(z,y)** $\wedge$ Réunion(y)

5.4 Réparation d'une requête

La réparation d'une requête conjonctive est le résultat d'un algorithme qui consiste à réparer successivement tous les conflits répertoriés sur la requête.
Si la requête initiale $Q(\bar{X})$ ne présente qu'un conflit, $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$, on obtient directement ses réparations $Q'_i(\bar{X})$ en remplaçant, dans $Q(\bar{X})$, Q_1 par chacune de ses réparations. Si la requête présente plusieurs conflits, on intègre dans la requête les réparations concernant le premier conflit, puis on teste si le deuxième conflit $(Q_2, \mathcal{D}_2, \mathcal{C}_2)$ est encore présent dans les $Q'_i(\bar{X})$. Ce test se fait en calculant un **saturé partiel** de chaque $Q'_i(\bar{X})$ en n'utilisant que les connaissances exprimées dans $\mathcal{D}_2$ et $\mathcal{C}_2$ puis en vérifiant si toutes les contraintes de $\mathcal{C}_2$ sont encore appliquées dans le saturé considéré. Si l'une des contraintes de $\mathcal{C}_2$ n'est plus appliquée cela signifie que le deuxième conflit n'existe plus après la réparation du premier. On peut donc réparer l'éventuel conflit suivant. Sinon, il faut effectuer de nouvelles réparations adaptées à ce deuxième conflit, puis traiter les conflits suivants.

Réparer une disjonction de requêtes conjonctives consiste à réparer tous les conflits répertoriés sur chacune des requêtes de la disjonction. Une optimisation future consistera à chercher si toutes les requêtes ont des conflits communs et à essayer de proposer des réparations globales.

On peut montrer que les réparations d'une requête ne présentent plus de conflit et ont chacune un généralisé commun avec la requête initiale. Elles ne sont plus nécessairement des généralisations de la requête initiale du fait des réparations par typage.

Proposition 5.2: *Soit $Q(\bar{X})$ une requête insatisfiable avec $\mathcal{D} \cup \mathcal{C} \cup \mathcal{D_V} \cup \mathcal{C_V}$. Soit $Q'(\bar{X})$ une réparation de $Q(\bar{X})$, on a les propriétés suivantes :*
$Q'(\bar{X}), \mathcal{D} \cup \mathcal{C} \cup \mathcal{D_V} \cup \mathcal{C_V} \not\models \perp$ et il existe $Q''(\bar{X})$ telle que $Q(\bar{X}), \mathcal{D} \cup \mathcal{D_V} \models Q''(\bar{X})$ et $Q'(\bar{X}), \mathcal{D} \cup \mathcal{D_V} \models Q''(\bar{X})$.

5.5 Exemple d'une requête insatisfiable du fait des sources disponibles.

Supposons qu'on ne dispose que des trois sources suivantes : les deux premières proposent des hôtels en Méditerranée ou aux Caraïbes, la dernière fournit des campings à La Réunion.

S1	r_{v11}	$V_{11}(x) \Rightarrow Hôtel(x)$
	r_{v12}	$V_{12}(x,y) \Rightarrow Localisé(x,y)$
	c_{v11}	$V_{12}(x,y) \wedge \neg V_{11}(x) \Rightarrow \perp$
	c_{v12}	$V_{11}(x) \wedge V_{12}(x,y) \wedge \neg EnMéditerrannée(y) \Rightarrow \perp$
S2	r_{v21}	$V_{21}(x) \Rightarrow Hôtel(x)$
	r_{v22}	$V_{22}(x,y) \Rightarrow Localisé(x,y)$
	c_{v21}	$V_{22}(x,y) \wedge \neg V_{21}(x) \Rightarrow \perp$
	c_{v22}	$V_{21}(x) \wedge V_{22}(x,y) \wedge \neg Caraïbes(y) \Rightarrow \perp$
S3	r_{v31}	$V_{31}(x) \Rightarrow Camping(x)$
	r_{v32}	$V_{32}(x,y) \Rightarrow Localisé(x,y)$
	c_{v31}	$V_{32}(x,y) \wedge \neg V_{31}(x) \Rightarrow \perp$
	c_{v32}	$V_{31}(x) \wedge V_{32}(x,y) \wedge \neg Réunion(y) \Rightarrow \perp$

En plus des connaissances décrivant la hiérarchie des lieux géographiques, on dispose des connaissances suivantes :

r_1 : $Hôtel(x) \Rightarrow LieuDeRésidence(x)$
r_2 : $Camping(x) \Rightarrow LieuDeRésidence(x)$
c_1 : $Camping(x) \wedge Hôtel(x) \Rightarrow \perp$
c_2 : $Réunion(x) \wedge Caraïbes(x) \Rightarrow \perp$
c_3 : $Réunion(x) \wedge EnMéditerrannée(x) \Rightarrow \perp$

Considérons la requête d'un utilisateur souhaitant obtenir un hôtel à La Réunion : Q(x) = Hôtel(x) $\wedge$ Localisé(x, y) $\wedge$ Réunion(y).

Le médiateur transmet les réécritures suivantes, insatisfiables vis à vis des connaissances précédentes :

1. $V_{11}(x) \wedge V_{12}(x,y) \wedge Réunion(y)$
2. $V_{21}(x) \wedge V_{22}(x,y) \wedge Réunion(y)$
3. $Hôtel(x) \wedge V_{32}(x,y)$.

Après saturation, chaque réécriture présente un conflit, respectivement :

1. $\{V_{11}(x), V_{12}(x,y), Réunion(y)\}, \{\}, \{c_{v12}, c_3\}$
2. $\{V_{21}(x), V_{22}(x,y), Réunion(y)\}, \{\}, \{c_{v22}, c_2\}$
3. $\{Hôtel(x), V_{32}(x,y)\}, \{\}, \{c_{v31}, r_{v31}, c_1\}$.

Les deux premiers conflits se réparant de la même façon, on obtient les deux réparations suivantes :

1. $Hôtel(x) \wedge Localisé(x,y) \wedge AuSoleil(y) \wedge ÎleAvecPlage(y)$,
2. $LieuDeRésidence(x) \wedge Localisé(x,y) \wedge Réunion(y)$.

6 Conclusion et perspectives

Notre objectif est d'aider un utilisateur à reformuler sa requête lorsque celle-ci n'obtient pas de réponse. Notre contribution a consisté à définir un cadre formel permettant de caractériser les causes minimales d'insatisfiabilité d'une requête en terme de conflits et à proposer une méthode de construction de réparation basée sur la généralisation de concept.
Ce travail est en cours d'implémentation en Java.

Rechercher toutes les causes minimales d'insatisfiabilité d'une requête et trouver toutes ses réparations sont des problèmes NP-difficiles, qui peuvent devenir polynomiaux lorsqu'on ne cherche pas à être exhaustif [5].

La recherche des sous-requêtes qui échouent présente des similarités avec la recherche des composants supposés défaillants dans les sytèmes de diagnostic de Reiter [10]. Les ensembles conflictuels de Reiter révèlent les composants ne pouvant être simultanément normaux. Les nôtres comportent de même les atomes de la requête initiale dont la présence conjointe crée une incohérence, mais contiennent de plus les règles intervenant dans la déduction de cette incohérence. Reiter rétablit la consistance de ces ensembles en supposant défaillant un composant par ensemble (cf. méthode des Hittings Sets) ce qui reviendrait, dans notre cas, à réparer la requête en supprimant les atomes considérés. Telle quelle, cette méthode ne prend donc pas en compte la possibilité de généraliser un atome en utilisant une règle et ne permet pas de retrouver toutes les réparations.

De plus, nous nous distinguons des travaux dans les domaines du Diagnostic par le formalisme que nous considérons et par l'exploitation différenciée des connaissances suivant leur nature. Nos contraintes pouvant contenir plusieurs littéraux négatifs, notre formalisme possède un pouvoir d'expression plus élevé que les formalismes classiquement considérés en Diagnostic. De même, en distinguant formellement parmi les contraintes, les sous-ensembles qui jouent des rôles différents dans la modélisation (contraintes de typage, d'exclusion, etc.) nous pouvons ensuite réutiliser ces distinctions pour guider la construction des réparations.

Les travaux les plus proches des nôtres sont, de fait, ceux développés dans le cadre des Bases de Données Déductives, par Motro [8] et les chercheurs de l'équipe de Minker [2], [3], [4] et [5].
Ainsi, Gal [4], présente à l'utilisateur les contraintes d'intégrité du domaine éventuellement violées par sa requête, sans proposer de réparations. Ne disposant pas à proprement parler de mécanisme d'inférences, ses contraintes ne portent que sur des prédicats extensionnels ou évaluables, ne contiennent au plus qu'un littéral négatif et toutes les dérivations possibles de l'ensemble de contraintes initiales doivent être préalablement générées comme de nouvelles contraintes pour prendre en compte leurs intéractions éventuelles. De même, la requête initiale doit être réécrite sous la forme

de prédicats extensionnels avant d'être confrontée à l'ensemble des contraintes d'intégrité précompilé. Du fait de la construction des contraintes dérivées, plusieurs contraintes sont souvent mises en cause sur une même requête. Gal doit donc effectuer une phase de tri guidée par des heuristiques pour ne présenter que les contraintes pertinentes.

Sans non plus construire de réparation, Motro cherche à identifier les sous-requêtes minimales qui échouent, que ce soit par rapport aux contraintes du domaine ou par rapport aux données présentes dans la base. Ces sous-requêtes sont calculées en testant toutes les généralisations immédiates de la requête initiale et en itérant ce processus sur toutes les généralisations encore en échec. Les sous-requêtes minimales en échec sont celles dont toutes les généralisations réussissent. Les généralisations d'une requête sont effectuées soit en supprimant un de ses littéraux, soit en faisant varier un paramètre suivant une échelle de valeurs prédéfinie. Cette méthode, comme celle de Reiter, permet d'éviter notre phase préalable de détection des conflits par saturation, mais nécessite en revanche de très nombreuses interrogations effectives de la base de données (ou appels à un démonstrateur chez Reiter). Ces interrogations nous sont inutiles quand l'échec est dû aux viols de contraintes ou à l'absence de sources satisfaisant la requête, puisque nous disposons d'une description abstraite des contenus des différentes sources de données disponibles.

L'objectif de Gaasterland [2] est d'élargir la requête, que ce soit pour obtenir des informations supplémentaires susceptibles d'intéresser l'utilisateur ou pour la réparer si elle n'a pas de réponses. Elle utilise pour ce faire un ensemble de règles décrivant la façon de relaxer chaque prédicat et éventuellement les constantes arguments de ces prédicats. Plusieurs directions de relaxation étant possibles pour un prédicat, les relaxations sont générées en largeur d'abord, par étapes, et soumises à l'utilisateur pour que celui-ci puisse choisir la direction qu'il préfère voir développer. Si la requête n'avait pas de réponses, Gaasterland réutilise le module d'identification des contraintes violées developpé par Gal, pour proposer en premier les relaxations permettant de corriger la requête.

Le cadre formel présenté ici n'est qu'un point de départ que nous comptons étendre pour prendre en compte la richesse du formalisme de CARIN [9] langage utilisé dans le projet PICSEL et qui combine des règles et des Logiques de Description (terminologiques).

De nombreuses optimisations devront aussi être introduites :

- dans la phase de saturation de la requête par les contraintes, pour éviter la création de trop nombreux conflits non pertinents,

- dans la phase de réparation, pour prendre en compte globalement les conflits et calculer les réparations minimales. Dans cette optique, une adaptation des techniques de Reiter, tenant compte du fait que les conflits ne sont pas uniquement dûs aux littéraux mais également aux connaissances qu'on leur applique, est en cours d'étude.

Remerciements

Nous remercions sincèrement C. Froidevaux et M.-C. Rousset pour leurs relectures attentives et nos fructueuses discussions.

Références

[1] Y. Arens, C. A. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6, 1996.

[2] T. Gaasterland. Cooperative Answering through Controlled Query Relaxation. *IEEE Expert*, pp. 48-59, sept-oct 1997.

[3] T. Gaasterland, P. Godfrey, and J. Minker. An overview of Cooperative Answering . *Journal of Intelligent Information Systems*, vol.1, pp. 123-157, 1992.

[4] A. Gal. Construction de réponses coopératives à l'aide de contraintes d'intégrité dans des bases de données logiques. *thèse de l'Université Rennes I*, décembre 1988.

[5] P. Godfrey. Minimization in cooperative response to failing databases queries. In *International Journal of Cooperative Information Systems (IJCIS)* 6(2) : 95-149 june 97.

[6] V. Lattes and M.-C. Rousset. The use of CARIN language and algorithms for Information Integration: the PICSEL project. In *Proceedings of the workshop "Intelligent Information Integration" associated with the European Conference on Artificial Intelligence (ECAI-98)*, Brighton, United Kingdom, August 98.

[7] A. Levy, A. Rajamaran, and J. Ordille. Query-answering algorithms for information agents. In *Proceedings of the thirtenth AAAI conference on Artificial Intelligence: AAAI'96*, 1996.

[8] A. Motro. Intensional Answers to Database Queries. *IEEE*, 6(3), pp 444-454, juin 1994.

[9] A. Levy and M.-C. Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence Journal*, 104, pp 165-209, 1998.

[10] R. Reiter. A Theory of Diagnosis from First Principles. In *Artificial Intelligence* 32, pp. 57-95, 1987.