# Repairing Queries in a Mediator Approach

**Alain Bidault, Christine Froidevaux and Brigitte Safar**[1]

**Abstract.** In this paper, we study unsatisfiable queries posed to a mediator in an information integration system and expressed in the logical formalism of the information integration system PICSEL[2]. First, we characterise *conflicts* as the minimal causes of the unsatisfiability of a query. Then, we produce its set of *repairs*: a repair is a query that does not generate any *conflict* and that has a common generalisation with the initial query and is semantically close to it.

## 1 Introduction

In recent years, the problem of information integration has received a lot of attention. In particular, several information integration systems (e.g., Information Manifold [10], PICSEL [9], SIMS [1]) have been based on a *mediator* architecture which provides a *uniform* query interface to multiple and possibly heterogeneous data sources. Users pose queries in terms of a set of relations designed to capture the semantics of a given application domain (e.g., tourism[3]). Those relations are *virtual* in the sense that their instances are not directly available but stored in the sources. As a consequence, answering a query means translating a user's query into a query that refers directly to the relevant sources, which needs a set of *source descriptions*. Our sources are described by a set of *views*, for which logical constraints and a logical mapping with domain relations are specified.

The most important advantage of a mediator is that it enables users to focus on specifying their demand, by freeing them from having to find the relevant sources and possibly combine data from multiple sources to obtain answers. Users do not have to know which sources are available. Instead, the mediator takes control of the construction of the specialised query plans (expressed in terms of views) to be executed in order to answer the original queries (expressed in terms of the domain model).

In the setting of data integration systems, the need for a *cooperative* query answering process is especially crucial because users do not know the contents of the data sources that are available. In particular, it may happen that the user's query, while being meaningful w.r.t the domain model, has no answer because its translation leads to specialised query plans that violate the constraints specifying the actual contents of the sources. In this case, it is important to *explain* to the user why his query failed. For instance, he asked for hotels located in England, and the only sources connected to the mediator provide hotels located in Germany. In addition, it is very useful to offer him a new query, called a *repair,* which is semantically close to the initial one and for which the mediator can provide answers. For instance, the user could be interested in a source providing Bed&Breakfasts in England, instead of hotels.

[1] L.R.I., C.N.R.S & University of Paris-Sud Bâtiment 490, 91405, Orsay Cedex, France {bidault, chris, safar}@lri.fr

[2] PICSEL is supported by the CNET (Centre National d'Études des Télécommunications ) under contract number 97 1B 378.

[3] in collaboration with a travel agency, see http://www.degriftour.fr/

In this paper, we consider the problem of repairing queries which do not obtain any answer, due to a violation of constraints. We take a logical framework (see section 2) for representing the domain model and the source description, associated with inference algorithms. These algorithms are the basis for computing the specialised query plans, and for checking their compatibility w.r.t the constraints.

Our contribution is twofold. First, as described in section 3, we characterise the minimal causes of the absence of any answer in terms of *conflicts.* Conflicts group together the rules and constraints responsible of the query's failure given the domain theory and the source descriptions. Second, as described in section 4, we build a set of repairs, such that each repair has a common generalisation with the initial query. We show, with examples coming from the tourism domain, several problems arising in the mediation context.

## 2 Representation of Domain, Sources and Queries

First, we specify the logical framework, called the domain theory. We describe the formalism used to express knowledge, the contents of the sources and queries. In the following, $\bar{X}_1, \ldots, \bar{X}_n$ and $\bar{Y}$ are tuples of variables.

### 2.1 Domain Knowledge

The knowledge domain is expressed by means of a declarative representation of object classes (Country, Flight, Stay, Travel...) and of relations among these classes. The domain is described using atoms, of the form $p(\bar{X})$ where $p$ is a relation name and $\bar{X}$ a tuple of variables or constants. We distinguish some unary relations, called *concepts*. They represent object classes relevant to the application domain. $C(x)$ is called an *atom-concept* if $C$ is a concept.
The domain knowledge $(\mathcal{D}, \mathcal{C})$ contains two components:

- **A set** $\mathcal{D}$, composed by rules of the form:

$p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$, where $q(\bar{Y})$ and $p_1(\bar{X}_1) \ldots p_n(\bar{X}_n)$ are atoms, and each variable in $\bar{Y}$ appears in $\bar{X}_1 \cup \ldots \cup \bar{X}_n$.

**ex:** A flight via Berlin is a flight having a stop in Berlin.
Flight(x) ∧ Stop(x, y) ∧ Berlin(y) ⇒ FlightViaBerlin(x).

We distinguish two kinds of rules:

$\mathcal{D}_h$ describes a *hierarchy* between the domain concepts, of the form: $C_1(x) \Rightarrow C_2(x)$, where $C_1$ and $C_2$ are concepts.

**ex:** A direct flight is a flight. DirectFlight(X) ⇒ Flight(X)

$\mathcal{D}_t$ describes *type-rules* of the form $p(\bar{X}) \Rightarrow C(x_i)$ with $x_i \in \bar{X}$. They specify which concept $C$ will characterise the arguments of $p(\bar{X})$. **ex:** the binary relation CityDeparture takes as a first argument travels and as a second argument geographical places.

CityDeparture(x,y) ⇒ Travel(x) CityDeparture(x,y) ⇒ GeoPlace(y)

Only $\mathcal{D}_h$ and $\mathcal{D}_t$ can have a concept as a conclusion.

Concepts and relations not appearing in rules as conclusions, except in type-rules, are called *base relations*.

- **A set** $\mathcal{C}$, composed by Horn constraints of the form:
$c : l_1(\bar{X}_1) \wedge \ldots \wedge l_n(\bar{X}_n) \Rightarrow \bot.$

**ex:** There is no NiceSwimming if there is no beach:
NoBeachPlace(x) ∧ FeasibleLeisure(x, y) ∧ NiceSwimming(y) ⇒ ⊥

The Semantics used are standard first order logic semantics. An interpretation $\mathcal{I}$ contains a non-empty domain $\mathcal{O}^{\mathcal{I}}$. It assigns an object $a^{\mathcal{I}} \in \mathcal{O}^{\mathcal{I}}$ to every constant $a$, and a relation of arity n over the domain $\mathcal{O}^{\mathcal{I}}$ to each relation of arity $n$.

- $\mathcal{I}$ is a model of a rule $r$ if whenever $\alpha$ is a mapping from the variables of the rule $r$ to elements of $\mathcal{O}^{\mathcal{I}}$, such that $\alpha(\bar{X}_i) \in {p_i}^{\mathcal{I}}$ for every $p_i$, we have $\alpha(\bar{Y}) \in q^{\mathcal{I}}$.

- $\mathcal{I}$ is a model of a constraint $c$ if for every assignment $\alpha$ of the variables of $c$ with elements from the domain $\mathcal{O}^{\mathcal{I}}$, we do not have simultaneously for every $l_i$: $\alpha(\bar{X}_i) \in {l_i}^{\mathcal{I}}$.

- $\mathcal{I}$ is a model of $(\mathcal{D}, \mathcal{C})$ if it is a model of each of its components.

## 2.2 Description of the Sources

The contents of a source $\mathcal{S}_i$ are represented using a vocabulary $\mathcal{V}$ constituted by as many local relations $v_{ij}$, called *views*, as we know the source $\mathcal{S}_i$ gives instances of domain base relations. The description of sources in terms of views contains two components:

- **A logical set of implications** $\mathcal{D}_v$**,** linking each view to a domain relation, $v_i(\bar{X}) \Rightarrow p(\bar{X})$.

**ex:** $\mathcal{S}_1$ provides instances of the concept *Hotel* and of the binary relation *Located.* $V_{11}(x) \Rightarrow$ Hotel(x) $V_{12}(x,y) \Rightarrow$ Located(x, y)

- **A set of Horn constraints** $\mathcal{C}_v$ characterising the view instances: $l_1(\bar{X}_1) \wedge \ldots \wedge l_n(\bar{X}_n) \Rightarrow \bot$, where $l_1 \ldots l_n$ are base relations and/or view names or their negation with at most one negation.

**ex:** instances from $V_{12}$ are linked to $V_{11}$ and hotels from $\mathcal{S}_1$ are located in the Caribbean : $V_{12}(x,y) \wedge \neg V_{11}(x) \Rightarrow \bot$
$V_{11}(x) \wedge Located(x,y) \wedge \neg InCaribbean(y) \Rightarrow \bot$

The semantics for $\mathcal{D}_v$ and $\mathcal{C}_v$ are the same as for $(\mathcal{D}, \mathcal{C})$.

## 2.3 Position of the Problem

We perform conjunctive queries of the form:
$Q(\bar{X}) : p_1(\bar{X}_1, \bar{Y}_1, \bar{a}_1) \wedge \ldots \wedge p_n(\bar{X}_n, \bar{Y}_n, \bar{a}_n)$
where $p_i$ are domain relation names appearing in $\mathcal{D} \cup \mathcal{C}$. The variables $\bar{X} = \bigcup_{i=1}^{n} X_i$ are called *distinguished variables* of the query and represent data, instances expected by the user when posing the query. $\bar{Y} = \bigcup_{i=1}^{n} Y_i$ are non-distinguished variables, and $\bar{a}_1, \ldots, \bar{a}_n$ are tuples of constants. The variables are existentially quantified.
Classically, answering a query $Q(\bar{X})$ is interpreted relatively to a database $\mathcal{DB}$, possibly associated to a domain theory $\mathcal{D}$, and consists in determining whether:
$\mathcal{DB}, \mathcal{D} \models \exists \bar{X},\ \exists \bar{Y}\, [p_1(\bar{X}_1, \bar{Y}_1, \bar{a}_1) \wedge \ldots \wedge p_n(\bar{X}_n, \bar{Y}_n, \bar{a}_n)].$
In the case of a positive answer, the *answer* to the query is the set of tuples $\bar{b} = \bar{b}_1 \cup \ldots \cup \bar{b}_n$ such that:
$\mathcal{DB}, \mathcal{D} \models \exists \bar{Y}\, [p_1(\bar{b}_1, \bar{Y}_1, \bar{a}_1) \wedge \ldots \wedge p_n(\bar{b}_n, \bar{Y}_n, \bar{a}_n)]$
In fact, in our mediator approach, we do not access the data source contents. Answering a query consists in searching the different expansions of this query in terms of views. An *expansion* of a query $Q(\bar{X})$ is a query that, using the domain theory $\mathcal{T} = \mathcal{D} \cup \mathcal{D}_{\mathcal{V}} \cup \mathcal{C} \cup \mathcal{C}_{\mathcal{V}}$, logically implies $Q(\bar{X})$, and that is satisfiable with $\mathcal{T}$.
We assume that $\mathcal{T} \not\models \bot$, $\mathcal{T} \not\models \forall \bar{X} \neg p(\bar{X})$, p a relation name, and that the dependency graph of the relations appearing in $\mathcal{T}$ is acyclic. Moreover, we limit ourselves to queries having no constant.

In the PICSEL [9] mediator context, an algorithm that determines expansions in terms of views has been developed and implemented in Java. This algorithm proceeds, using $\mathcal{D}$ and $\mathcal{D}_{\mathcal{V}}$, by successive rewritings $Q_{\mathcal{R}}(\bar{X})$ of the initial query $Q(\bar{X})$ in a backward chaining way. We have $Q_{\mathcal{R}}(\bar{X}), \mathcal{D} \cup \mathcal{D}_{\mathcal{V}} \models Q(\bar{X})$.

These rewritings are the nodes of a tree rooted by Q. While developing the tree, for each node $Q_{\mathcal{R}}$, the satisfiability with $\mathcal{T}$ is tested. A node is a leaf either if $Q_{\mathcal{R}}(\bar{X}), \mathcal{T} \models \bot$, or if $Q_{\mathcal{R}}$ can no longer be rewritten.

Our problem arises when all the leaves are unsatisfiable with $\mathcal{T}$. Our aim is to obtain repairs of $Q(\bar{X})$, satisfiable with $\mathcal{T}$, and, most of the time, that are a least satisfiable generalisation of $Q(\bar{X})$.

It is worth noticing that $Q_{\mathcal{R}}(\bar{X})$ can simply be the initial query, detected as unsatisfiable before any expansion step.

**Example 1:** Assume that we have the following two sources: the first one offers hotels in the Mediterranean, the second one provides campsites in Reunion.

| | | |
|---|---|---|
| S1: | $r_{v11}$ | $V_{11}(x) \Rightarrow Hotel(x)$ |
| | $r_{v12}$ | $V_{12}(x,y) \Rightarrow Located(x,y)$ |
| | $c_{v11}$ | $V_{12}(x,y) \wedge \neg V_{11}(x) \Rightarrow \bot$ |
| | $c_{v12}$ | $V_{11}(x) \wedge Located(x,y) \wedge \neg InMediterranean(y) \Rightarrow \bot$ |
| S2: | $r_{v21}$ | $V_{21}(x) \Rightarrow Campsite(x)$ |
| | $r_{v22}$ | $V_{22}(x,y) \Rightarrow Located(x,y)$ |
| | $c_{v21}$ | $V_{22}(x,y) \wedge \neg V_{21}(x) \Rightarrow \bot$ |
| | $c_{v22}$ | $V_{21}(x) \wedge Located(x,y) \wedge \neg Reunion(y) \Rightarrow \bot$ |
| $\mathcal{C}$ | $c_1$ : | $Campsite(x) \wedge Hotel(x) \Rightarrow \bot$ |
| | $c_2$ : | $Reunion(x) \wedge InMediterranean(x) \Rightarrow \bot$ |
| $\mathcal{D}$ | $r_0$ : | $Campsite(x) \Rightarrow ResidencePlace(x)$ |

Suppose that the user wishes to get an hotel in Reunion:
Q(x) = Hotel(x) ∧ Located(x, y) ∧ Reunion(y).
The mediator provides the following unsatisfiable rewritings:
$Q_{\mathcal{R}_1}$ $Hotel(x) \wedge V_{12}(x,y) \wedge Reunion(y)$
$Q_{\mathcal{R}_2}$ $Hotel(x) \wedge V_{22}(x,y) \wedge Reunion(y)$.

Our task is to identify the origins of the unsatisfiablity of the given queries and to propose satisfiable repairs of the initial query Q. In order to do that, queries $Q_{\mathcal{R}}$ given by PICSEL are saturated with $\mathcal{T}$ to identify their inconsistency, which is expressed in terms of conflicts (section 3). How to repair these conflicts is described in section 4.

# 3 Definition of a Conflict

We note $Q_1$ a subset of a conjunctive query $Q$, $\mathcal{D}_1$ a subset of $\mathcal{D} \cup \mathcal{D}_v$, and $\mathcal{C}_1$ a subset of $\mathcal{C} \cup \mathcal{C}_v$.

**Definition 3.1:** *A* **conflict** *for a query Q is a triplet* $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *such that* $Q_1, \mathcal{D}_1, \mathcal{C}_1 \models \bot$. *$(\mathcal{D}_1, \mathcal{C}_1)$ is the* **cause** *of the conflict.*

We distinguish conflicts according to two criteria. The first one concerns the query's atoms appearing in the conflict.

**Definition 3.2:** *A conflict* $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *is* **Q_minimal** *if there does not exist* $Q_2 \subset Q_1$ *(strictly) such that* $Q_2, \mathcal{D}_1, \mathcal{C}_1 \models \bot$

**Example 2:**
$\mathcal{C}$ $c_1$: d(x) ∧ e(x) ∧ f(x) ⇒ ⊥ $c_2$: a(x) ∧ b(x) ⇒ ⊥
$\mathcal{D}$ $r_1$: a(x) ⇒ d(x) $r_2$: b(x) ⇒ e(x) $r_3$: c(x) ⇒ f(x)
$Q(x)$: a(x) ∧ b(x) ∧ c(x)
$Cf_1$: ({a(x), b(x), c(x)},{},{$c_2$}) $Cf_3$: ({a(x), b(x)},{},{$c_2$})
$Cf_2$: ({a(x), b(x), c(x) },{$r_1, r_2, r_3$},{$c_1$}
Because $\{a(x), b(x)\} \subset \{a(x), b(x), c(x)\}$, $Cf_1$ is not Q_minimal.

The second criterium concerns the knowledge appearing in the conflict. Thus, we define an operator on the causes of conflicts.

**Definition 3.3:** *The* **strict inclusion operator on causes** *is defined as follows:* $(\mathcal{D}_1, \mathcal{C}_1) \subset (\mathcal{D}_2, \mathcal{C}_2)$ *if and only if*

$$(\mathcal{D}_1 \subset \mathcal{D}_2 \text{ and } \mathcal{C}_1 \subseteq \mathcal{C}_2) \text{ or } (\mathcal{D}_1 \subseteq \mathcal{D}_2 \text{ and } \mathcal{C}_1 \subset \mathcal{C}_2).$$

**Definition 3.4:** *A* **minimal cause** *of conflict for an atoms conjunction* $Q_1$ *is a couple* $(\mathcal{D}_1, \mathcal{C}_1)$ *such that there exists no cause* $(\mathcal{D}_2, \mathcal{C}_2) \subset (\mathcal{D}_1, \mathcal{C}_1)$ *with* $Q_1, \mathcal{D}_2, \mathcal{C}_2 \models \perp$.

¿From definitions 3.2 and 3.4, we distinguish the relevant conflicts from among all the detectable ones available.

**Definition 3.5:** *A conflict* $(Q_1, \mathcal{D}_1, \mathcal{C}_1)$ *is* **relevant** *if it is Q_minimal and if* $(\mathcal{D}_1, \mathcal{C}_1)$ *is a minimal cause of conflict for* $Q_1$.

**Example 3:**

| | | |
|---|---|---|
| $\mathcal{D}_h$: | $r_1$ father(x) $\Rightarrow$ male(x) | $r_2$ mother(x) $\Rightarrow$ female(x) |
| | $r_3$ father(x) $\Rightarrow$ adult(x) | $r_4$ mother(x) $\Rightarrow$ adult(x) |
| $\mathcal{C}$: | $c_1$ male(x) $\wedge$ female(x) $\Rightarrow \perp$ | $c_2$ adult(x) $\wedge$ child(x) $\Rightarrow \perp$ |
| Q(x) = | father(x) $\wedge$ mother(x) $\wedge$ child(x) | |
| *Relevant conflicts* | cf1. ({father(x), mother(x)},{$r_1$,$r_2$},{$c_1$}) | |
| | cf2. ({father(x), child(x)},{$r_3$},{$c_2$}) | |
| | cf3. ({mother(x), child(x)},{$r_4$},{$c_2$}) | |

Complete and correct methods, as the positive hyperresolution method defined in [2], let us detect all the relevant conflicts generated by a given query, focusing on the literals of the query. Conflicts could be presented to the user as a first and rough explanation of his query's failure.

# 4 Repairs

Let us consider an unsatisfiable query $Q(\bar{X})$. We have to determine a set of repairs without conflicts. A repair is a satisfiable query $Q'(\bar{X})$ such that $Q(\bar{X})$ and $Q'(\bar{X})$ have a close common generalisation. Our repairs are based on the notion of concept subsumption.

## 4.1 Concept Generalisation

**Definition 4.1:** *A* **direct subsumer** *of a concept* $\mathcal{C}$ *is a concept* $\mathcal{C}'$ *such that the rule* $\mathcal{C}(x) \Rightarrow \mathcal{C}'(x)$ *is in* $\mathcal{D}_h$.

A concept can have many direct subsumers.

**Definition 4.2:** $\mathcal{C}'$ *is a* **subsumer** *of* $\mathcal{C}$ *if* $\mathcal{C}'$ *is a direct subsumer of* $\mathcal{C}$ *or if there exists* $\mathcal{C}''$ *such that* $\mathcal{C}'$ *is a direct subsumer of* $\mathcal{C}''$ *and* $\mathcal{C}''$ *is a subsumer of* $\mathcal{C}$.

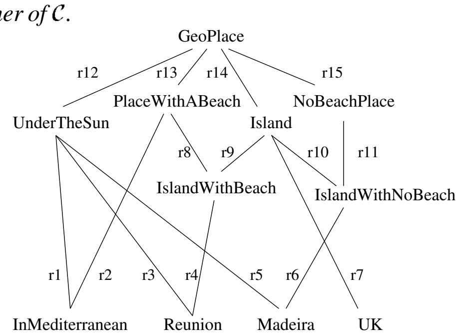


Figure 1 Geographicals Places Hierarchy

Figure 1 presents a fragment of the concept hierarchy $\mathcal{D}_h$. The direct subsumers of *Madeira* are *UnderTheSun* and *IslandWithNoBeach* while other subsumers are *Island, NoBeachPlace* and *GeoPlace*.

Type-rules specify for each argument of the relation, the concept $C_g$ to which it could belong. A *relevant query* verifies type-rules if it only uses atom-concepts more specific than or equal to $C_g$.

**Definition 4.3:** *Let* $Q(\bar{X}) = \bigwedge_{i=1}^{n} \mathcal{A}_i(\bar{X}) \wedge \bigwedge_{j=1}^{m} \mathcal{C}_j(\bar{X})$, *with* $\mathcal{A}_{i\,1 \leq i \leq n}$ *a n_ary relation and* $\mathcal{C}_{j\,1 \leq j \leq m}$ *a concept.* $Q(\bar{X})$ *verifies a type-rule* $r_t : \mathcal{A}_i(\bar{X}) \Rightarrow \mathcal{C}'(x)$, *if and only if the concept* $\mathcal{C}'$ *is one of the concepts* $\mathcal{C}_j$ *or a subsumer of one of the* $\mathcal{C}_j$*'s.*

**Example 4:** In addition to rules expressed in figure 1, suppose that Located has two type-rules and that Hotel is a place for residence:

$r_{16}$: Located(x, y) $\Rightarrow$ ResidencePlace(x) $r_{17}$: Located(x, y) $\Rightarrow$ GeoPlace(y)
$r_{18} \in \mathcal{D}_h$ : Hotel(x) $\Rightarrow$ ResidencePlace(x)

The query Q(x) = Hotel(x) $\wedge$ Located(x,y) $\wedge$ Madeira(y) verifies the type-rules associated to the relation $Located$.

The following definition establishes the notion of generalisation of an atom-concept $C(x)$.

**Definition 4.4:** *Let* $\mathcal{C}$ *be a concept and* $\mathcal{C}_1$, $\mathcal{C}_2$,...,$\mathcal{C}_n$, *its direct subsumers, the* **direct generalisation** $\mathcal{G}_c(x)$ *of the* **atom-concept** $\mathcal{C}(x)$ *is the conjunction* $\mathcal{C}_1(x) \wedge \mathcal{C}_2(x) \wedge ... \wedge \mathcal{C}_n(x)$.

**Remarks** (i) We use the conjunction of direct subsumers in order to stay close to the concept, taking into account all its features.
(ii) We have the following property: $\mathcal{C}(x), \mathcal{D}_h \models \mathcal{G}_c(x)$.

Thanks to the previous definitions, a generalisation of an atom-concepts conjunction will be defined.

**Definition 4.5:** *A* **direct generalisation** $\mathcal{G}_d(x)$ *of an* **atom-concepts conjunction** $\mathcal{C}_1(x) \wedge \ldots \wedge \mathcal{C}_{i-1}(x) \wedge \mathcal{C}_i(x) \wedge \mathcal{C}_{i+1}(x) \wedge \ldots \wedge \mathcal{C}_n(x)$ *is a conjunction of the form* $\mathcal{C}_1(x) \wedge \ldots \wedge \mathcal{C}_{i-1}(x) \wedge \mathcal{G}_{ci}(x) \wedge \mathcal{C}_{i+1}(x) \wedge \ldots \wedge \mathcal{C}_n(x), 1 \leq i \leq n$, *where* $\mathcal{G}_{ci}(x)$ *is the direct generalisation of* $\mathcal{C}_i(x)$.

**Definition 4.6:** $\mathcal{G}(x)$ *is a* **generalisation** *of an atom-concepts conjunction* $\mathcal{C}_{ac}(x)$ *if* $\mathcal{G}(x)$ *is a direct generalisation of* $\mathcal{C}_{ac}(x)$ *or if there exists an atom-concepts conjunction* $\mathcal{G}'(x)$ *such that* $\mathcal{G}(x)$ *is a direct generalisation of* $\mathcal{G}'(x)$ *and* $\mathcal{G}'(x)$ *is a generalisation of* $\mathcal{C}_{ac}(x)$.

**Example 4:** A generalisation of $InMediterranean(x) \wedge Island(x)$ is UnderTheSun(x) $\wedge$ PlaceWithABeach(x) $\wedge$ Island(x).

## 4.2 Query Repairs

In this section, we show how to use generalisations to repair a conflict in a query. We first define the notion of unsolvable query.

**Definition 4.7:** *A literals conjunction* $Q(\bar{X})$ *is* **unsolvable** *if it is unsatisfiable, that is,* $Q \cup \mathcal{D} \cup \mathcal{C} \models \perp$, *or if all the leaves* $Q_{\mathcal{R}}$ *of its rewriting tree are unsatisfiable with* $\mathcal{T}$, *that is,* $Q_{\mathcal{R}} \cup \mathcal{T} \models \perp$.

**Definition 4.8:** *Let* $Q(\bar{X}) = \mathcal{C}_{ac}(x) \wedge q(\bar{X})$ *be an unsolvable query, where* $\mathcal{C}_{ac}(x)$ *is a conjunction of atom-concepts, and* $q(\bar{X})$ *a conjunction of literals with* $x \in \bar{X}$. $\mathcal{R}(\bar{X}) = \mathcal{C}'_{ac}(x) \wedge q(\bar{X})$ *is a* **repair** *of* $Q(\bar{X})$ *if* $\mathcal{C}'_{ac}(x)$ *is a generalisation of* $\mathcal{C}_{ac}(x)$ *and if* $\mathcal{R}(\bar{X})$ *is not unsolvable.*

**Remark** We have: $Q(\bar{X}), \mathcal{D}_h \models \mathcal{R}(\bar{X})$. As repairs $\mathcal{R}(\bar{X})$ do not always verify the type-rules, the notion of relevant repair is needed.

**Definition 4.9:** *A repair* $\mathcal{R}(\bar{X})$ *of a query* $Q(\bar{X})$ *is a* **relevant repair** *if it verifies the type-rules associated to its own n-ary relations.*

Then, as many repairs could pretend to satisfy the previous definitions, only minimal relevant repairs have to be given to the user.

**Definition 4.10:** *$\mathcal{R}(\bar{X})$ is a* **minimal repair** *of the unsolvable query $Q(\bar{X})$ if $\mathcal{R}(\bar{X})$ is a relevant repair of $Q(\bar{X})$ and if there does not exist $R(\bar{X})$, a relevant repair of $Q(\bar{X})$, such that $R(\bar{X}), \mathcal{D}_h \models \mathcal{R}(\bar{X})$.*

**Example 4 continued:** we add the following knowledge:
$r_{19}$: NiceSwimming(x) ⇒ Leisure(x) $r_{20}$: AssLeisure(x, y) ⇒Leisure(y)
$r_{21}$: AssLeisure(x, y) ⇒ ResidencePlace(x)
$c_1$: NoBeachPlace(x) ∧ FeasibleLeisure(x,y) ∧ NiceSwimming(y) ⇒ ⊥
$c_2$: ResidencePlace(x) ∧ AssLeisure(x,z) ∧ Located(x,y) ∧
GeoPlace(y) ∧ ¬ FeasibleLeisure(y, z) ⇒ ⊥
and the query, Q(x) = Hotel(x) ∧ Located(x,y) ∧ Madeira(y) ∧ AssLeisure(x,z) ∧ NiceSwimming(z).

We identify a relevant conflict: ({Located(x,y), Madeira(y), AssLeisure(x,z), NiceSwimming(z)}, $\{r_6, r_{11}, r_{15}, r_{20}\}$, $\{c_1, c_2\}$).

Substituting in $Q(x)$ the atom-concept *Madeira(y)* by its generalisation, *Island(y)* ∧ *UnderTheSun(y)*, a minimal repair $Q_{\mathcal{R}}$(x) is found: Hotel(x) ∧ Located(x,y) ∧ **Island(y)** ∧ **UnderTheSun(y)** ∧ AssLeisure(x,z) ∧ NiceSwimming(z).

## 4.3 Algorithms for Calculating Repairs

In [14], Reiter proposes an algorithm to determinate a minimal set of abnormal components of a system, by computing minimal hitting sets for the collection of conflicts set. We need to adapt his method. Instead of simply deleting the literals, we want to generalise them, if it is possible, taking into account $\mathcal{D}_h$. We construct the minimal repairs from the set of all minimal conflicts. Repairing a query means repairing each of its conflict, and for this, we calculate the set of atom-rules pairs *ar-pairs* that need to be oversteped.

**Definition 4.11:** *Let $cf_1 = (Q_1, D_1, C_1)$ be a relevant conflict. We define as many* **ar-pairs** *$a - r_h$, with $a \in Q_1$ and $r_h \subseteq D_1 \cap \mathcal{D}_h$, as there exist maximal paths starting from $a$ and associated to $r_h$, developed during the hyperresolution process.*

**Definition 4.12:** *The* **overstep generalisation** *$\mathcal{G}_{a-r_h}(x)$, is the generalisation of* a *where all the concepts reached by applying rules of $r_h$ have been successively replaced by their direct generalisations.*

**Remark** Note that, for a given conflict, oversteping one of its ar-pairs does not *always* solve the conflict.

**Example 3 continued:** the sets of ar-pairs for each relevant conflict are: ar1= { father-$r_1$, mother-$r_2$}, ar2= {father-$r_3$, child}, ar3= {mother-$r_4$, child}. Having the following rule, $r_5$ adult(x) ⇒ $\geq 10Years$(x), the generalisation oversteping father- $r_1$ is $adult(x)$ and the one for father-$r_3$ is $male(x) \wedge \geq 10Years(x)$.

**Definition 4.13:** *Let $Q(\bar{X}) = \bigwedge_{i=1}^{n} a_i(x_i) \wedge \bigwedge_{j=1}^{m} q_j(\bar{Y})$ be a query where $\bar{X} \subset \bigcup_{i=1}^{n} x_i \cup \bar{Y}$ and the $a_i(x_i)$'s are atom-concepts. Let $a_i - r_{h\,i}$ be ar-pairs, for $1 \leq i \leq n$. The* **query oversteping all** $a_i$**-**$r_{h\,i}$ **in** *$Q(\bar{X})$ is $\bigwedge_{i=1}^{n} \mathcal{G}_{a_i - r_{h\,i}}(x_i) \wedge \bigwedge_{j=1}^{m} q_j(\bar{Y})$.*

Given an unsolvable query $Q$, and given a set of relevant conflicts, our aim is to obtain the **minimal** sets of ar-pairs that solve all the conflicts for $Q$ together. These sets of pairs are given as leaves of an Order-Minimal-Repair-Tree. The following algorithm constructs this *OMR-tree ;* its input is the set of relevant conflicts for $Q$ and a total order $\mathcal{O}_p$ over ar-pairs for avoiding redundant results.

**Definition of a Total Order $\mathcal{O}_p$ Over Ar-pairs**
The order follows first the user's preferences over atoms, by default, the order of the literals in the query. For pairs having the same literal, the order follows the decreasing number of rules $|r_h|$. Then, the lexicographical order over the rule names is followed.

**Calculation of an *OMR-Tree***
Let CS be a set of relevant conflicts. An edge-labeled and node-labeled tree T is an *OMR-tree* for CS iff it is a smallest tree with the following properties:

- Its root is labeled by "$\surd$" if CS is empty, otherwise by CS.
- If $n$ is a node of T, define $EL(n)$ to be the set of edge labels on the path in T from the root node to $n$.
- The label for a node is the collection of conflicts that have not been treated while oversteping the ar-pairs $EL(n)$, if such a collection exists. Otherwise, $n$ is labeled by $\surd$.
- If $n$ is closed or labeled by $\surd$, it is a leaf of T.
- Nodes are generated breadth-first and edges are listed from left to right according to the increasing order $\mathcal{O}_p$ on edges' labels.
- Let $n$ be a node labeled by $\Sigma \subseteq CS$, and $n_{\surd}$ a node of T labeled by $\surd$. If $EL(n_{\surd})$ is a subset of $EL(n)$, $n$ is closed.
- If $n$ is the root, for each ar-pair $ar$ associated to $\Sigma$, $n$ has a successor node $n_{ar}$. Otherwise, n is issued from a previous edge labeled $ar_p$, and it has, for each ar-pair $ar$ greater than $ar_p$, a successor node $n_{ar}$. Each node $n_{ar}$ is linked to $n$ by an edge labeled by $ar$.
- If $ar = a\text{-}r_{h\,k}$ is such that there exist ar-pairs $\{a\text{-}r_{h\,1}, \ldots, a\text{-}r_{h\,k}\}$ in $EL(n_{ar})$, where $\bigcup_{i=1}^{k} r_{h\,i} = \bigcup_{i=1}^{k} r_{h\,i} \backslash r_{h\,j}$, for some $j \in [1..k]$, then $n_{ar}$ is closed.

**Remark** Such a tree lets us first find a repair close to the query in terms of generalisations steps and quickly get a repair close to the user's preferences ($\mathcal{O}_p$).

**Theorem 1** *Let Q be an unsolvable query and let $T$ be the* OMR-tree *associated to it. By oversteping in Q all ar-pairs $EL(l)$ given by one leaf $l$ labelled by "$\surd$" in $T$, we get a minimal repair of the query. Conversely, every minimal repair can be obtained in this way.*

**Example 3 continued:** five ar-pairs are calculated:
*child, mother-$r_2$, mother-$r_4$, father-$r_1$, father-$r_3$.*

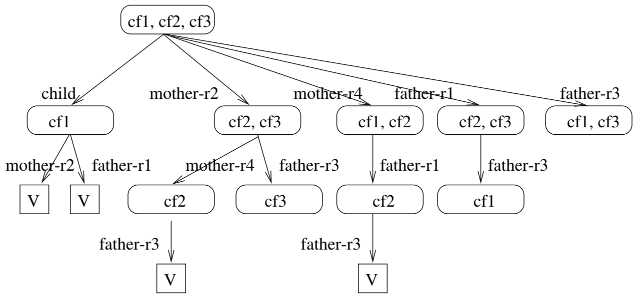


Figure 2. OMR-tree for example 3

Figure 2 presents the 4 minimal sets of ar-pairs to be oversteped:

| | |
|---|---|
| 1. child, mother-$r_2$ | 3. mother-$\{r_2, r_4\}$, father-$r_3$ |
| 2. child, father-$r_1$ | 4. mother-$r_4$, father-$\{r_1, r_3\}$. |

Thus, the four following minimal repairs can be proposed:

| | |
|---|---|
| 1. father(x) | 3. male(x) ∧ $\geq 10Years(x) \wedge$ child(x) |
| 2. mother(x) | 4. female(x) ∧ $\geq 10Years(x) \wedge$ child(x). |

These repairs are obtained by oversteping the pairs computed using the OMR-tree. For example, (1.) is obtained removing child which has no direct subsumer, and by replacing mother by adult, which is not kept because it subsumes father.

### 4.4 Other Query Repairs

We give some indications of how to define new repairs since repairing using concept generalisation is not always quite satisfactory.

• When repairs do not verify the ***type-rules,*** we also have to modify their concerned binary atoms $R(x, y)$. For $Q(x) = C_1(x) \wedge R(x, y) \wedge C_2(y)$ which does not verify the type-rule $r_1 : R(x, y) \Rightarrow C_3(x)$, because of the exclusion-constraint $c_1 : C_1(x) \wedge C_3(x) \Rightarrow \perp$, two kinds of repairs (called ***type-repairs***) are possible, depending on whether we favor the binary relation or the concepts:
- the first one keeps the atom R(x, y) and consists in correctly typing the associated concept, replacing $C_1(x)$ by $C_3(x)$.
- the second one keeps the concepts and consists in replacing the relation $R$ by an atoms chain which correctly links $C_1(x)$ to $C_2(y)$. The research of such a chain needs to study the binary relations associated to the considered concepts or their subsumers. The new repair must verify the type-rules associated to its new binary relations. We are restricted to chains having at most two relations.

**Example 4 continued :** with the added following knowledge,
$r_{22}$ : FeasibleLeisure(x, y) $\Rightarrow$ GeoPlace(x)
$c_3$ : ResidencePlace(x) $\wedge$ GeoPlace(x) $\Rightarrow \perp$
Suppose the user's query is:
Q(x) = $Hotel(x) \wedge FeasibleLeisure(x, y) \wedge NiceSwimming(y)$

This query is incompatible with the domain structure because a hotel is a ResidencePlace whereas FeasibleLeisure needs a GeoPlace. Moreover, the repair using concept generalisations that modify the concept $Hotel$ cannot verify $r_{22}$.
The type-repairs are as follows : favoring FeasibleLeisure:
$\mathcal{R}_1(x)$ : **GeoPlace(x)** $\wedge$ FeasibleLeisure(x, y) $\wedge$ NiceSwimming(y),
favoring Hotel and NiceSwimming:
$\mathcal{R}_2(x)$ : Hotel(x) $\wedge$ **AssLeisure(x, y)** $\wedge$ NiceSwimming(y).

• We continue **Example 1**, described in section 2.3, to present the problems encountered because of the **sources**. In addition, the domain theory contains the rules expressed in figure 1.
Each rewriting contains a conflict, respectively :
$Q_{\mathcal{R}_1} : cf_1 \quad (\{V_{12}(x, y), Reunion(y)\}, \{r_{v12}\}, \{c_{v11}, c_{v12}, c_2\})$
$Q_{\mathcal{R}_2} : cf_2 \quad (\{Hotel(x), V_{22}(x, y)\}, \{r_{v21}, r_{v23}\}, \{c_1\})$.

Two repairs should be found: $Hotel(x) \wedge Located(x, y) \wedge UnderTheSun(y) \wedge IslandWithBeach(y)$, for the first conflict; $ResidencePlace(x) \wedge Located(x, y) \wedge Reunion(y)$ for the second.

## 5 Conclusion and Perspectives

Our objective is to help a user to reformulate his query detected as unsatisfiable. This paper has presented a formal framework that lets us characterise the minimal causes of the query's unsatisfiability in terms of conflicts. It has introduced the notion of minimal repairs of a query by means of concept generalisations. Moreover, an algorithm to calculate these repairs has been proposed, and is being implemented in Java.

Our study is related to work done in diagnosis [14] but differs as we have identified different subsets in the domain theory (type rules, source descriptions, etc.), so that we can give the user significant explanations for why his query failed, and propose repairs that meet his requirements, as closely as possible.

Due to a lack of place, we *succinctly* present related research developed in a deductive databases context.

Motro [11] introduces minimal failing sub-queries (MFS) that fail because of the domain constraints or the real data. He modifies the query by deleting literals or by relaxing, to a certain degree, some of its conditions. He does not have to detect all the conflicts beforehand, but, instead, he must often consult the effective data of the sources.

Godfrey [8] shows that looking for all the MFS and finding all their repairs are NP-hard problems, which can become polynomial when exhaustiveness is not necessary.

We were interested in an algorithm that gives some outputs, known as minimal, without having to wait for all the solutions to be calculated first, that is, our *OMR-Tree* algorithm.

Gal [7] gives the user integrity constraints that have been violated by a query, as we do, but she does not offer any repairs. Gaasterland [5][6] modifies the query either to get more information, which is relevant for the user, or to repair the query if it fails. She describes how to relaxe each predicate and its constant arguments. As there are many possibilities, relaxations are generated in breadth first, and at each level, they are submitted to the user, who has to decide, interactively, which relaxation he prefers.

The work we have presented should be extended in order to take into account the full expressiveness of the formalism CARIN [13], language used in the PICSEL project and which combines Horn rules with description logics. It could be interesting, first, to go further in exploiting the analogies with diagnosis, trying to map other algorithms [3], [4], and second, to investigate the links between our notion of generalisation and the generality quasiorders used in the Inductive Logic Programming field [12]. On the other hand, we have to formalise other methods to obtain repairs when generalisations are not quite satisfactory, that is, when the atoms of the query are badly typed or when the available sources are not relevant for the user. At last, some optimisations could be introduced, (i) to order and group the repairs according to their meaning, (ii) to avoid calculating irrelevant conflicts, generated by the type-rules for example, during the hyperresolution process.

## REFERENCES


[1] Y. Arens, C. A. Knoblock and W.-M. Shen. Query reformulation for dynamic information integration. *J. Intelligent Information Syst.*, 6, 96.

[2] Chang and Lee. Symbolic Logic and Mechanical Theorem Proving, 100-121, Academic Press, 73.

[3] L. Console and O. Dressler. Model-based diagnosis in the real world: lessons learned and challenges remaining. In *IJCAI'99*,1393-1400, 99.

[4] O. Dressler and P. Struss. Model-based Diagnosis with the Default-based Diagnosis Engine: Effective Control Strategies that Work in Practice. In *ECAI'94*, 677-681, 94.

[5] T. Gaasterland. Cooperative Answering through Controlled Query Relaxation. *IEEE Expert*, 48-59, sept-oct 97.

[6] Gaasterland, Godfrey, Minker. An overview of Cooperative Answering. *Journal of Intelligent Information Systems*, (1) 123-157, 92.

[7] A. Gal. Cooperative Responses in Deductive Databases. *PhD thesis, Departement of Computer Science, University of Maryland*, 88.

[8] P. Godfrey. Minimization in cooperative response to failing databases queries. In *Int. J. of Cooperative Information Syst.* 6(2): 95-149, 97.

[9] F. Goasdoué, V. Lattes and M.-C. Rousset. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL Project. In *Int. J. of Cooperative Information Syst. 99.*

[10] A. Levy, A. Rajamaran, and J. Ordille. Query-answering algorithms for information agents. In *Proc AAAI'96*, 40-47, august 96.

[11] A. Motro. Intensional Answers to Database Queries. In *IEEE*, 6(3), 444-454, june 94.

[12] S-H. Nienhuys-Cheng and R. de Wolf, Foundations of Inductive Logic Programming, *LNAI*, 1228, Springer Verlag 97.

[13] A. Levy and M.-C. Rousset. Combining Horn Rules and Description Logics in CARIN. In *Artificial Intelligence*, 104, 165-209, 98.

[14] R. Reiter. A Theory of Diagnosis from First Principles. In *Artificial Intelligence*, 32, 57-95, 87.