THE USE OF CARIN LANGUAGE AND ALGORITHMS FOR INFORMATION INTEGRATION: THE PICSEL SYSTEM

François Goasdoué, Véronique Lattès and Marie-Christine Rousset
L.R.I. Computer Science Laboratory
C.N.R.S & University of Paris-Sud
Building 490, 91405, Orsay Cedex, France
{goasdoue,mcr}@lri.lri.fr

PICSEL is an information integration system[1] over sources that are distributed and possibly heteregeneous. The approach which has been chosen in PICSEL is to define an information server as a knowledge-based mediator in which CARIN is used as the core logical formalism to represent both the domain of application and the contents of information sources relevant to that domain. In this paper, we describe the way the expressive power of the CARIN language is exploited in the PICSEL information integration system, while maintaining the decidability of query answering. We illustrate it on examples coming from the tourism domain, which is the first real case that we have to consider in PICSEL, in collaboration with the travel agency Degriftour[2].

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 [11], TSIMMIS [7], SIMS [2], Infomaster [10]) have been proposed, based on a *knowledge-based mediator* architecture. A mediator is an interface between users and existing information sources (that are distributed and possibly heteregeneous), which gives its users the illusion of a centralized and homogeneous information system. It allows them to ask domain-level queries and takes in charge in their place the access to the relevant sources in order to obtain the answers to their queries.

Most of knowledge-based mediator systems have in common the use of a logical formalism to describe both the *domain model* and the *contents of the information sources*. They differ, first, by the specific logical formalism that they use, second, by the way they express the contents of the information sources relatively to the domain model.

[1] granted by CNET (Centre National d'Etudes des Telecommunications) under contract number 97 1B 378

[2] see http://www.degriftour.fr/

As the relational model is widespread in databases, most information integration systems using a logic-based technology are based on function-free Horn rules (i.e Datalog). However, the advantages of Description Logics (DL) for information integration have also been pointed out ([3, 4]). DL's have been specially designed for modeling and reasoning on complex data descriptions, and their expressive power is well-suited for a natural conceptual modeling of the domain and of the sources. DL's deal with unary relations (referred to as *concepts*), representing sets of objects, and binary relations (referred to as *roles*). They vary according to the *constructors* they allow for defining complex concepts and roles. They are associated with inference algorithms that automatically structure the set of concepts, based on the subsumption relations existing between pairs of concepts. When DL's are viewed as a query language, concept subsumption corresponds to query containment. In fact, the restriction of DL's to unary and binary relations yields new cases for which containment is decidable, whereas it is undecidable for datalog. Recently, containment algorithms for a query language combining datalog and DL's, CARIN, have been developed [12]. Decidability results on query containment have also been obtained ([5]), for a DL extended with n-ary relations, $\mathcal{DLR}$.

As for the description of the sources, two approaches have been distinguished in the literature ([16, 9]), depending on the mapping between the domain relations (called the *global relations*) and the sources relations (called the *local relations*) which represent the content of the information sources. In the *Local as view* approach, illustrated by Information Manifold [11] or Infomaster [10], the source relations are defined as *views* over domain relations. In the *Global as view* approach, illustrated by TSIMMIS [7], the mapping is done from the domain relations to the source relations: the source relations are considered as new base relations in term of which some global relations can be expressed. The two approaches have dual advantages and drawbacks. The main advantage of the *Local as view* approach is its expressivity and flexibility for representing the contents of the sources relatively to an application domain. Its drawback is the complexity of the problem of answering queries. While in the *Global as view* approach, when queries are acyclic datalog rules, answering a query can be done by a simple unfolding of the rules, in the *Local as view* approach, the problem of answering queries becomes a problem of reformulation of queries in terms of views. This problem of *rewriting queries using views* has been studied for several classes of relational queries and views ([16]). It is undecidable when queries are recursive ([8]). In [3], it has been shown that the problem of rewriting queries using views, when views and queries are conjonctive queries over the $\mathcal{ALN}$ description logic, may be undecidable except if some drastic restrictions are imposed on the query and on variables appearing in the views.

In this paper, we describe the way the expressive power of the CARIN language is exploited in the PICSEL information integration system, while maintaining the decidability of query answering. The main characteristics of our approach can be summarized as follows.

• For modeling the domain and expressing queries, our approach benefits from combining the expressive powers of function-free Horn rules and the $\mathcal{ALN}$ description logics. Although $\mathcal{ALN}$ is a rather restricted DL, this provides a rich modeling framework, compared to the pure relational approaches. In addition, this modeling framework is equipped with the reasoning services that have been designed for CARIN [12] , namely concept and rule subsumption.

• For describing the contents of the information sources, we impose a limitation for computational reasons. As a matter of fact, it has been pointed out in [3], that if we apply a full *Local as view* approach in the setting of $\mathcal{ALN}$-CARIN the decidability of query answering is not guaranteed. In the setting of PICSEL, the only views that we allow are (possibly complex) $\mathcal{ALN}$ expressions over domain concepts and roles. More precisely, the content of an information source is described by a set of source relations associated with:

-a mapping that relates each source relation to a domain relation, indicating, at the domain level, the type of data that can be extracted from the source,

-a set of (integrity and terminological) constraints that enable fine-tuning the characterization of the actual data that can be obtained from the source.

The mapping between the source relations and the domain relations follows a *Global as view* approach. However, the (terminological) constraints allow the expression of some kinds of views (i.e, expressible by $\mathcal{ALN}$ expressions). As a result, our approach combines a *Global as view* approach with a restricted *Local as view* approach. In the setting of CARIN, where views and queries are expressed using DL expressions combined with datalog rules, this limitation is not so big from an expressive power point of view, compared with pure relational approaches, while guaranteing that query rewriting is decidable.

• The queries that are handled by PICSEL are posed in terms of domain relations and have the form of non recursive $\mathcal{ALN}$-CARIN rules, i.e, unions of conjunctive queries over $\mathcal{ALN}$ expressions. Query expansion in $\mathcal{ALN}$-CARIN is the core algorithmic tool for query evaluation in PICSEL. Expanding the query consists of computing a representative set of all the possible rewritings of the initial query in terms of source relations. In the setting of non recursive function-free Horn rules, building query expansions can be simply done by unfolding the rules. In the setting of DL, when the query is a single concept-atom, and the source relations are all mapped with concepts, query expansion can be reduced to subsumption checking. However, the problem of expanding (unions of) conjunctive queries into (unions of) conjunctions of role-atoms and concept-atoms has not been addressed so far.

The main algorithmic contribution of this paper is that we provide a sound and complete algorithm for query expansion in the $\mathcal{ALN}$ description logic and in $\mathcal{ALN}$-CARIN. As a result, we obtain a new class of queries and views for which the problem of rewriting queries using views is decidable.

The paper is organized as follows. Section 2 summarizes the use of CARIN as the core logical formalism of PICSEL. Section 3 describes query processing in PICSEL. Finally, section 4 relates PICSEL to some existing integration information systems.

2. Logical representation of the domain and of the sources

In PICSEL, CARIN is used to represent both the domain of application and the contents of information sources that are available and relevant to that domain.

We illustrate it on examples coming from the tourism domain, which is the first real case that we have to consider in PICSEL, in collaboration with the Web (and Minitel) travel agent Degriftour [3]. This travel agent makes available on-line three databases that contain different types of touristic products (flights, tours, stays in different places), each of them having its own specificities. For example, the *BonjourFrance* database offers a large variety of touristic products but all of them are located in France. The so-called *Degriftour* database offers flights, stays and tours for a lot of destinations all over the world. Its specificity however is that all its offers correspond to a departure date which is within the next two weeks. As a counterpart, the corresponding prices are specially interesting. The *Reductour* database provides rather similar products but with a less strong constraint on the departure date: it just has to be within the next eight months. Other differences exist between the contents of those three databases. For instance, we might know that the only housing places that are proposed in the *Reductour* database are hotels, while the others can provide rooms in Bed&Breakfast in addition to hotel rooms.

2.1. Syntax and semantics of CARIN

A CARIN knowledge base (KB) contains two components: a set $\mathcal{R}$ of rules, and a set $\mathcal{T}$ of Description Logics statements. Description logics statements are definition or inclusion statements about concepts and roles in the domain. Concepts and roles are unary and binary relations that can also appear in the antecedents of the rules. Relations that do not appear in the description logics statements are called *ordinary relations*. Ordinary relations can be of any arity.

Description Logics component in CARIN**:** The DL component of a CARIN knowledge base in PICSEL contains *concept definitions* and some kinds of *concept inclusions*, using the $\mathcal{ALN}$ DL. $\mathcal{ALN}$ contains the DL constructors of conjunction ($C_1 \sqcap C_2$), value restriction ($\forall R.C$), number restrictions $((\geq\ n\,R), (\leq\ n\,R))$, and negation ($\neg A$) restricted to basic concepts only.

- Concept definitions are of the form $CN := ConceptExpression$, where CN is a concept name and $ConceptExpression$ is a concept expression. We assume that a concept name appears in the left hand side of at most one concept definition. *Atomic concepts* are those which do not appear in any left hand side of a definition. A concept name CN depends on a concept name CN' if CN' appears in the definition of CN. A set of concept definitions is said *acyclic* if there is no cycle in the concept names dependency relation. In the setting of PICSEL, we consider only acyclic concept definitions. We can *unfold* an acyclic set of definitions by iteratively substituting every concept name with its definition.

- The concept inclusions that are allowed in the PICSEL setting are of the form:

  - $A \sqsubseteq ConceptExpression$, where A is a atomic concept,

[3]see http://www.degriftour.fr/

– or $A_1 \sqcap A_2 \sqsubseteq \perp$, where A_1 and A_2 are atomic concepts.

We consider that every concept is unfolded and put in a normal form of a conjunction of concept expressions of the (*simple*) form: A (atomic concept), $\neg A$, $(\geq n\, R)$, $(\leq n\, R)$, or of the (*complex*) form: $\forall R_1 \forall R_2 \ldots \forall R_k.D$, where D is simple.

Rule component in CARIN: The rule component $\mathcal{R}$ of a CARIN knowledge base contains a set of rules that are logical sentences of the form:

$$\forall \bar{X}[p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})]$$

where $\bar{X}_1, \ldots, \bar{X}_n, \bar{Y}$ are tuples of variables (included in $\bar{X}$) or constants. We require that the rules are safe, i.e., a variable that appears in $\bar{Y}$ must also appear in $\bar{X}_1 \cup \ldots \cup \bar{X}_n$. As a shortcut, in the following, the variable quantification will be omitted. The relations $p_1, \ldots, p_n$ may be either concept names or expressions, role names, or ordinary relations that do not appear in $\mathcal{T}$. The relation q must be an ordinary relation.

The *base* relations are those which do not appear in any consequent of rules. In particular, any concept or role appearing in the rules are base relations. We call a base atom an atom $p(\bar{X})$ where p is a base relation. We call *concept-atom* an atom $p(X)$ where p is a concept name or expression, and *role-atom* an atom $r(X, Y)$ where r is a role name.

An ordinary relation p is said to *depend* on an ordinary relation q if q appears in the antecedent of a Horn rule whose consequent is p. A set of rules is said to be *recursive* if there is a cycle in the dependency relation among ordinary relations. In the setting of PICSEL, we consider non recursive rules.

We can consider [4] that all the concept-atoms appearing in the rules are of one of the forms: $A(X)$, $\neg A(X)$, $(\geq n\, R)(X)$, $(\leq n\, R)(X)$, or $\forall R_1 \forall R_2 \ldots \forall R_k.D(X)$, where D is a simple concept.

Since the rules are safe, without loss of generality, we can assume that in every rule, we have the disequality $X \neq Y$ for every pair of distinct variables appearing in the rule. For clarity, we omit these atoms in our examples and algorithms.

Semantics of CARIN KBs: The semantics of CARIN KBs is the standard model-based semantics of first-order logic. An interpretation I contains a non-empty domain $\mathcal{O}^I$. It assigns to every constant a an object $\alpha^I(a) \in \mathcal{O}^I$, and a relation of arity n over the domain $\mathcal{O}^I$ to every relation of arity n. In particular, it assigns a unary relation C^I to every concept in $\mathcal{T}$, and a binary relation R^I over $\mathcal{O}^I \times \mathcal{O}^I$ to every role R in $\mathcal{T}$. The extensions of concept and role descriptions are given by the following equations: ($\sharp\{S\}$ denotes the cardinality of a set S):

$(C \sqcap D)^I = C^I \cap D^I$,
$(\neg A)^I = \mathcal{O}^I \setminus A^I$,

[4]If $E_1 \sqcap \ldots \sqcap E_n$ is the normal and unfolded form of C, we replace the atome $C(X)$ by the atom conjunction $E_1(X) \wedge \ldots \wedge E_n(X)$

$(\forall R.C)^I = \{d \in \mathcal{O}^I \mid \forall e : (d,e) \in R^I \rightarrow e \in C^I\}$
$(\geq n\, R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \geq n\}$
$(\leq n\, R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \leq n\}$

An interpretation I is a model of a knowledge base $(\mathcal{T}, \mathcal{R})$ if it is a model of each of its components $\mathcal{T}$ and $\mathcal{R}$. An interpretation I is a model of $\mathcal{T}$ if $A^I \subseteq D^I$ for every inclusion $A \sqsubseteq D$ in $\mathcal{T}$, $CN^I = D^I$ for every concept definition $CN := D$. If CE and DE are two concept expressions, we say that *CE is subsumed by DE* if $CE^I \subseteq DE^I$ in every interpretation I. An interpretation I is a model of a rule $r : p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$ if, whenever α is a mapping from the variables [5] of r to the domain $\mathcal{O}^I$, such that $\alpha(\bar{X}_i) \in p_i^I$ for every atom of the antecedent of r, then $\alpha(\bar{Y}) \in q^I$.

2.2. Modelling the domain of application

Our approach for modelling the domain is similar in spirit to most of the existing mediator approaches. We define a basic vocabulary in terms of names of base relations that are meaningful for the application domain (e.g., tourism), and we use our logical formalism (CARIN) for defining new relations that are significant for the tourism domain and that can be defined over the base relations. In the setting of CARIN there are two ways of defining complex relations: by rules, or by concept expressions. It is illustrated by the following example.

Example 2.1: Let us suppose that our basic vocabulary contains atomic concepts such as *HousingPlace*, *Breakfast*, *CollectiveBuilding*, *PrivateBuilding*, which respectively denote the set of housing places and varied sets of services or buildings, and roles such as *AssRoom*, *AssMeal* and *AssBuilding*, which denote binary relations between housing places and their associated buildings, room and meal services. We can define the two new concepts *Hotel* and *Bed&Breakfast* by the following DL definitions:

$$Hotel := HousingPlace \sqcap (\geq\ 5\, AssRoom) \sqcap (\forall AssBuilding.CollectiveBuilding)$$

$$\begin{aligned} Bed\&Breakfast := {} & HousingPlace \sqcap (\geq\ 1\, AssRoom) \sqcap (\geq\ 1\, AssMeal) \\ & \sqcap (\forall AssMeal.Breakfast) \sqcap (\forall AssBuilding.PrivateBuilding) \end{aligned}$$

$$CollectiveBuilding \sqcap PrivateBuilding \sqsubseteq \perp$$

Rules can be used to define new n-ary relations. For instance, the following rule defines the notion of flights combined with stays as a 4-ary relation *FlightStay*. A stay combined with a flight is characterized by a departure city (denoted by the first variable *Dcity* in the consequent of the rule), an arrival city (denoted by the variable *Acity*), a departure date (*Ddate*), a return date (*Rdate*). The possible combinations of a flight to and back a given destination with a stay at that place

[5] Distinct variables are mapped to distinct elements

obey some constraints that are expressed by the conditions in the antecedent of the rule.

$$
\begin{aligned}
& Stay(S) \wedge Flight(V) \wedge Assoc(S,H) \wedge Located(H, ACity) \\
& \wedge ArrivalCity(V, Acity) \wedge DepartureCity(V, Dcity) \wedge DepartureDate(V, Ddate) \\
& \wedge ReturnDate(V, Rdate) \wedge BeginningDate(S, Ddate) \wedge EndDate(S, Rdate) \\
& \quad \Rightarrow FlightStay(Dcity, Acity, Ddate, Rdate)
\end{aligned}
$$

2.3. *Modelling the contents of the information sources*

Our approach for describing the information sources has been guided by the necessity of trading off expressive power against decidadibility of query answering. More precisely, each information source $\mathcal{S}$ is characterized by a set of source relations $\mathcal{V}_S$, and described by a CARIN knowledge base (with its standard semantics, described in subsection 2) which contains:

(i) a set $\mathcal{I}_S$ of rules $v(\bar{X}) \Rightarrow p(\bar{X})$ that indicate which kind of data can be found in the source $\mathcal{S}$. The p's are domain relations and there are as many source relations v's in $\mathcal{V}_S$ (and as many implications in $\mathcal{I}_S$) as domain relations whose instances can be found in the source $\mathcal{S}$,

(ii) a set $\mathcal{C}_S$ of constraints on the instances of the source relations. We allow two types of constraints. *Terminological constraints* are of the form $v \sqsubseteq C$, where C is a concept expression. *Integrity constraints* are of the form $l_1(\bar{X}_1) \wedge \ldots \wedge l_n(\bar{X}_n) \Rightarrow \perp$ where the l_i's are source relations or negation of source relations.

A *source atom* is an atom of the form $v(\bar{X})$ where v is a source relation.

As an example, we can have the following (partial, for this illustration) descriptions of the three information sources that we have previously mentioned (i.e., $BonjourFrance$, $Degriftour$ and *Reductour*).

Example 2.2: The rules in the description of the three sources say that we can find instances of housing places and flights in all of them (together with instances of associated properties like their location, their departure cities and dates ...). The constraints contained in each description enable distinguishing between them: for instance, the housing places and flights that can be found in $BonjourFrance$ are restricted to be located in France; the housing places that can be found in *Reductour* are necessarily hotels. Some constraints serve to express that instances of some binary relations that can be found in a given source are exclusively related to some unary relations that can be found in the same source: for instance, the constraint $v^2_{BF}(X,Y) \wedge \neg v^1_{BF}(X) \Rightarrow \perp$ in $Descr(BonjourFrance)$ expresses that the locations that can be found in the source $BonjourFrance$ (denoted by the source relation v^2_{BF} and the implication $v^2_{BF}(X,Y) \Rightarrow Located(X,Y)$) are only locations that are related to the housing places that can be found in the same source (denoted by the source relation v^1_{BF} and the implication $v^1_{BF}(X) \Rightarrow HousingPlace(X)$).

$Descr(BonjourFrance) : v^1_{BF}, \ldots, v^n_{BF}$	
mappings: $\mathcal{I}_{BonjourFrance}$	constraints: $\mathcal{C}_{BonjourFrance}$
$v^1_{BF}(X) \Rightarrow HousingPlace(X)$	$v^1_{BF} \sqsubseteq (\forall Located.France)$
$v^2_{BF}(X,Y) \Rightarrow Located(X,Y)$	$v^2_{BF}(X,Y) \wedge \neg v^1_{BF}(X) \Rightarrow \bot$
$v^3_{BF}(X) \Rightarrow Flight(X)$	$v^3_{BF} \sqsubseteq (\forall ArrivalCity.France)$
$v^4_{BF}(X,Y) \Rightarrow ArrivalCity(X,Y)$	$v^4_{BF}(X,Y) \wedge \neg v^3_{BF}(X) \Rightarrow \bot$
...	...

$Descr(Degriftour) : v^1_D, \ldots, v^n_D$	
mappings: $\mathcal{I}_D$	constraints: $\mathcal{C}_D$
$v^1_D(X) \Rightarrow HousingPlace(X)$	$v^3_D \sqsubseteq (\forall DepartureDate.NextTwoWeeks)$
$v^2_D(X,Y) \Rightarrow Located(X,Y)$	
$v^3_D(X) \Rightarrow Flight(X)$	...
$v^4_D(X,Y) \Rightarrow ArrivalCity(X,Y)$	$v^4_D(X,Y) \wedge \neg v^3_D(X) \Rightarrow \bot$
...	...

$Descr(Reductour) : v^1_R, \ldots, v^n_R$	
mappings: $\mathcal{I}_R$	constraints: $\mathcal{C}_R$
$v^1_R(X) \Rightarrow HousingPlace(X)$	$v^1_R \sqsubseteq Hotel$
$v^2_R(X,Y) \Rightarrow Located(X,Y)$	
$v^3_R(X) \Rightarrow Flight(X)$	$v^3_R \sqsubseteq (\forall DepartureDate.NextEightMonths)$
$v^4_R(X,Y) \Rightarrow ArrivalCity(X,Y)$	$v^4_R(X,Y) \wedge \neg v^3_R(X) \Rightarrow \bot$
...	...

3. Query processing in PICSEL

The queries that we consider are unions of conjunctive queries over domain relations, of the form $Q(\bar{X}) : \bigvee_{i\in[1..k]} p^i_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge p^i_m(\bar{X}_m, \bar{Y}_m)$, where the p^i_j's are domain relations, some of which may be concept expressions or role names. The variables of $\bar{X} = \bar{X}_1 \cup \ldots \cup \bar{X}_m$ are called the *distinguished* variables of the query: they represent the variables of the query which the user is interested in knowing the instances when he asks the query. The variables that are not distinguished (denoted by $\bar{Y} = \bar{Y}_1 \cup \ldots \cup \bar{Y}_m$) are called *existential* variables of the query: they are existentially quantified and their use is to constrain the distinguished variables. For example, by the query $Q(X) : Hotel(X) \wedge Located(X,Y) \wedge France(Y)$, the user specifies that the answers he is interested in, are all the possible instances of X that are hotels, and for which there exists instances of Y (their locations) that are in France.

As for the rules, we can assume without loss of generality that disequalities $X \neq Y$ are implicit for every pair of distinct variables that appears in the query.

Classically, a query is interpreted relatively to a database *db*, which consists of a (finite) set of ground atoms representing a set of stored data, and possibly to a set $\mathcal{DT}$ of logical sentences representing intentional knowledge. Given a model $\mathcal{I}$ of *db* and $\mathcal{DT}$, a conjunctive query $Q_i(\bar{X}) : p^i_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge p^i_m(\bar{X}_m, \bar{Y}_m)$ over *db* and $\mathcal{DT}$ is interpreted as the set Q^I_i of tuples $\bar{o}$ made of elements of the interpretation domain $\mathcal{O}^I$, such that, when substituting $\bar{o}$ for $\bar{X}$, the formula $\exists \bar{Y}\, p^i_1(\bar{o}_1, \bar{Y}_1) \wedge$

$\ldots \wedge p^i_m(\bar{o}_m, \bar{Y}_m)$ evaluates to true in $\mathcal{I}$ [6]. A union of conjunctive query $Q(\bar{X})$: $Q_1(\bar{X}) \vee \ldots \vee Q_k(\bar{X})$ is interpreted as the set Q^I of tuples $\bar{o}$ made of elements of $\mathcal{O}^I$, such that the formula $Q(\bar{o})$ evaluates to true in $\mathcal{I}$.

The answer of a query $Q(\bar{X})$ over a database db and $\mathcal{DT}$ is the set of tuples $\bar{a}$ such that: $db, \mathcal{DT} \models \bigvee_{i\in[1..k]} \exists \bar{Y}\, p^i_1(\bar{a}_1, \bar{Y}_1) \wedge \ldots \wedge p^k_m(\bar{a}_m, \bar{Y}_m)$.

In the setting of PICSEL, the stored data db are not directly available in the mediator but are abstracted by the set of source relations $\mathcal{V}$. Answering queries in PICSEL resorts then to find conjunctions of source atoms (called *rewritings*) which entail the initial query. The intentional knowledge is composed of the domain model (a set of rules $\mathcal{R}$ and a terminology $\mathcal{T}$), and of the source descriptions (a set $\mathcal{I}_\mathcal{V}$ of logical implications and a set $\mathcal{C}_\mathcal{V}$ of constraints involving the source relations $\mathcal{V}$).

Definition 3.1: *Let $Q(\bar{X}) : \bigvee_{i\in[1..k]} p^i_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge p^i_m(\bar{X}_m, \bar{Y}_m)$ be a query expressed in term of domain relations, let $Q_\mathcal{V}(\bar{X}, \bar{Z})$ be a conjunction of source atoms. $Q_\mathcal{V}(\bar{X}, \bar{Z})$ is a rewriting of $Q(\bar{X})$ iff:*

$$\exists \bar{Z}\, Q_\mathcal{V}(\bar{X}, \bar{Z}), \mathcal{R} \cup \mathcal{T} \cup \mathcal{I}_\mathcal{V} \cup \mathcal{C}_\mathcal{V} \models \bigvee_{i\in[1..k]} \exists \bar{Y}\, p^i_1(\bar{a}_1, \bar{Y}_1) \wedge \ldots \wedge p^k_m(\bar{a}_m, \bar{Y}_m)$$

A rewriting of a query $Q(\bar{X})$ can then be viewed as a specialized query plan that can be directly executed on the sources, and which provides answers to the initial query. Query processing in PICSEL consists of computing a *representative* set of all the possible rewritings of the initial query, in order to get all the possible answers that can be obtained for it by interrogating the available sources. For doing so, given an initial query $Q(\bar{X}) : \bigvee_{i\in[1..k]} Q_i(\bar{X})$, we proceed in two main steps.

(i) **Expansion of each conjunctive query $Q_i(\bar{X})$ composing $Q(\bar{X})$:** This algorithmic stage is an iterative process based on successive steps of atom expansions. First, ordinary atoms are expanded by a standard backward-chaining unfolding of the rules $\mathcal{R}$. As a result, we obtain *ordinary expansions*. Then, (conjunctions of) concept-atoms and role-atoms of each ordinary expansion are expanded. As a result, for each conjunctive query $Q_i(\bar{X})$, we obtain a set of terminal expansions, which is representative of all the ways of deriving it from conjunctions of *terminal* atoms. Terminal atoms are either source atoms or base domain atoms that cannot be logically derived from source atoms. It is easy to see that terminal expansions made of source atoms only provide rewritings of the initial query $Q(\bar{X})$: since they entail the conjunctive query $Q_i(\bar{X})$, they entail the union $Q(\bar{X})$ too. It is more subtle to understand that terminal expansions whose some conjuncts are not source atoms may still provide valid rewritings of the initial query $Q(\bar{X})$. Let $v_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge v_i(\bar{X}_i, \bar{Y}_i) \wedge remainder(\bar{X}, \bar{Y})$ be a terminal expansion where $remainder(\bar{X}, \bar{Y})$ is a conjunction of terminal atoms that are not source atoms. If the remainder is only made of concept-atoms, it may be the case that $v_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge v_i(\bar{X}_i, \bar{Y}_i)$ by itself, while not entailing the conjuntive query $Q_i(\bar{X})$, still entails the union $Q(\bar{X})$. Intuitively, it is due to the fact that the

[6] Distinct variables are mapped to distinct elements

concept-atoms may introduce negation in our logical setting. Consequently, we have to consider as *candidate rewritings* the conjunctions of source atoms that are obtained by keeping only source atoms from the different terminal expansions.

(ii) **Verification of the candidate rewritings:** for each candidate rewriting, we have to check (a) whether it is compatible with the integrity constraints, and (b) whether, possibly by a case-analysis reasoning on the concept expressions appearing in the remainders, it actually entails the initial query. We use the *existential entailment* algorithm, which is described in [12], in order to check whether the candidate expansion is satisfiable and entails one of the remainders of the terminal expansions it comes from. It consists of first, constructing a set of completions from $v_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge v_i(\bar{X}_i, \bar{Y}_i)$, second evaluating the remainders on each completion. The construction of completions is based on using propagation rules that account for description logics constructors but also on simulating a case-analysis reasoning by injecting in the completions universal sentences of the form $\forall x[C(x) \vee \neg C(x)]$ for every simple concept expression C appearing in the remainders.

Before describing the main algorithmic steps of query processing in PICSEL, we first illustrate it through the following example. This example, though simple, enables us to point out some subtle points of reasoning with conjunctive sentences involving complex concept expressions together with binding variables over several conjuncts. In particular, it shows the necessity of the verification step. It also points out that conjunctions of concept-atoms and role-atoms can be entailed by *single* concept-atoms (called their *descriptive support*), if their existential variables satisfy a particular binding. Replacing such conjunctive sentences by their descriptive supports, before proceeding to the expansion of each concept-atom and role-atom, can be necessary to find rewritings that would not be found otherwise, while they do entail the query.

Example 3.1: Suppose that our basic vocabulary contains: (a) the atomic concepts *Flight* and *AmCity*, which respectively denote the set of flights, and the set of American cities ; (b) the role *Stop* which is a binary relation between flights and cities. Suppose that the domain model is described by the following set $\mathcal{R}$ of rules and the following terminology $\mathcal{T}$. The rules in $\mathcal{R}$ state two different ways for a flight to be convenient for American people:

$r_1 : Flight(X) \wedge (\leq 1\, Stop)(X) \Rightarrow ForAm(X)$

$r_2 : Flight(X) \wedge Stop(X, Y) \wedge AmCity(Y) \Rightarrow ForAm(X)$

Rule r_1 says that a flight which has at most one stop is convenient for American people. Rule r_2 says that a flight which has a stop in an American city is convenient for American people too.

The terminology $\mathcal{T}$ is reduced to a single definition stating that *AmFlight* is the set of flights that stops only in American cities:

$AmFlight := Flight \sqcap \forall Stop.AmCity$

Suppose that we have only one source relation v, and that its description is reduced to the mapping $\mathcal{I}$: $v(X) \Rightarrow AmFlight(X)$.

Consider the query $Q(X) : ForAm(X)$.

The ordinary expansions of $Q(X)$, resulting from rule unfolding, are:
$Q_1(X) : Flight(X) \wedge (\leq 1\, Stop)(X)$
$Q_2(X) : Flight(X) \wedge Stop(X, Y) \wedge AmCity(Y)$

The terminal expansions that are obtained by our expansion algorithm are:
$Exp_1 : v(X) \wedge (\leq 1\, Stop)(X)$
$Exp_2 : v(X) \wedge (\geq 1\, Stop)(X)$

Exp_1 is obtained by expanding the concept-atom $Flight(X)$ in the first ordinary expansion $Q_1(X)$. This expansion is supported by the presence of $AmFlight$ in $\mathcal{T}$, which is subsumed by $Flight$, and the implication in $\mathcal{I}$ mapping $v(X)$ to $AmFlight(X)$. As a result, $v(X)$ is obtained as a necessary and sufficient source atom for entailing the domain atom $Flight(X)$. The atom $(\leq 1\, Stop)(X)$ is terminal because it cannot be derived from source atoms.

Exp_2 is obtained from the second ordinary expansion $Q_2(X)$:

- first, by determining that $Flight(X) \wedge Stop(X, Y) \wedge AmCity(Y)$ has a descriptive support which is $[Flight \sqcap (\geq 1\, Stop) \sqcap (\forall Stop.AmCity)](X)$,

- second, by replacing $Q_2(X)$ by the conjunction of concept-atoms coming from its descriptive support: $Flight(X) \wedge (\geq 1\, Stop)(X) \wedge (\forall Stop.AmCity)(X)$,

- third, by expanding the concept-atoms $Flight(X)$ and $(\forall Stop.AmCity)(X)$.

Both terminal expansions contain terminal atoms that are not source atoms: $(\leq 1\, Stop)(X)$ in Exp_1, $(\geq 1\, Stop)(X)$ in Exp_2.
From those terminal expansions, we obtain the candidate rewriting $v(X)$.
Existential entailment is then used to check whether $v(X), \mathcal{T} \cup \mathcal{I} \models Exp_1 \vee Exp_2$. Case analysis reasoning is guided by the remainders: in the building of completions, one branching corresponds to adding $(\leq 1\, Stop)(X)$, the second one corresponding to adding its negation, i.e., $(\geq 2\, Stop)(X)$. In the completion obtained in the first branching, Exp_1 is evaluated to true, while in the second branching Exp_2 is evaluated to true. As a result, $v(X)$ is proved as being a valid rewriting of $Q(X)$.

We now focus on describing the query expansion process in PICSEL which is a new algorithmic contribution. The verification step uses algorithms that have been previously described in [12]. The expansion process applies to each conjunctive query composing the initial query. Expanding a conjunctive query consists of:

- computing a set of *ordinary expansions*, obtained by expanding the ordinary atoms of the query, as described in subsection 3.1,

- expanding conjunctions of concept-atoms and role-atoms appearing in each ordinary expansion by its descriptive support, when it is possible, in order to get *compact expansions*, as described in subsection 3.2,

- then, expanding each concept-atom and role-atom in the compact and ordinary expansions, in order to get a set of *terminal expansions*, as described in subsection 3.3.

3.1. *Expanding ordinary atoms*

Let $p(\bar{X})$ be an ordinary atom. It is expanded by iteratively unfolding the rules whose consequent is of the form $p(\bar{X}')$. Let $r : p_1(\bar{X}'_1, \bar{Y}'_1) \wedge \ldots \wedge p_k(\bar{X}'_k, \bar{Y}'_k) \Rightarrow p(\bar{X}')$ be a rule of $\mathcal{R}$. Let α be the most general unifier of $p(\bar{X})$ and $p(\bar{X}')$, extended such that every variable Y'_i is assigned to a fresh variable that appears nowhere else. An unfolding of the rule r consists of replacing in the query the atom $p(\bar{X})$ by the conjunction: $p_1(\alpha(\bar{X}'_1), \alpha(\bar{Y}'_1)) \wedge \ldots \wedge p_k(\alpha(\bar{X}'_k), \alpha(\bar{Y}'_k))$. A step of expansion of the atom $p(\bar{X})$ results in the set of conjunctions obtained by unfolding all the rules whose consequent is unifiable with $p(\bar{X})$. Since the rules are non recursive, we obtain, after a finite number of steps of ordinary atoms expansion, a set of conjunctions (that we call *ordinary expansions*) in which all the ordinary atoms are either source atoms or atoms that are not unifiable with any consequent of rule.

It results from the soundness and completeness of backward-chaining algorithm for non recursive function-free Horn rules that the set of expansions of an ordinary atom characterizes all the ways of deriving it from base atoms [7].

Let $CJ_{ord}(\bar{X}_1, \bar{Y}_1) \wedge CJ_{term}(\bar{X}_2, \bar{Y}_2)$ be an ordinary expansion. We distinguish its ordinary part $(CJ_{ord}(\bar{X}_1, \bar{Y}_1))$, composed of ordinary atoms, from its terminological part $(CJ_{term}(\bar{X}_2, \bar{Y}_2))$ which is a conjunction of concept-atoms and role-atoms. Its distinguished variables are $\bar{X}_1 \cup \bar{X}_2$ and its existential variables are $\bar{Y}_1 \cup \bar{Y}_2$. As previously seen, the different (distinguished or existential) variables are considered as distinct. We now focus on $CJ_{term}(\bar{X}_2, \bar{Y}_2)$. First, we determine whether it is possible to obtain *compact* expansions of it by replacing conjunctions of concept-atoms and role-atoms by their descriptive supports. Then, we proceed to the expansion of each concept-atom and each role-atom appearing in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$ or its compact expansions.

3.2. *Determining the descriptive support of a conjunction of concept-atoms and role-atoms*

Let Z and Z' be two variables appearing in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$. We say that Z' is a direct successor of Z in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$ if there exists an atom $R(Z, Z')$ in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$. The successor relationship is the transitive closure of the direct successor relationship. For every $X \in \bar{X}_2$, let $s(X)$ be the set of all the successors of X that are existential variables. Let $cj(X)$ be the conjunction of all the atoms of $CJ_{term}(\bar{X}_2, \bar{Y}_2)$ which are concept-atoms involving X or concept-atoms and role-atoms involving the variables in $s(X)$. We define the graph G accounting for the binding of the variables appearing in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$ as follows: the nodes in the graph are the variables, and there is an arc from any variable to any of its direct successor. We say that $cj(X)$ has a *tree structure* iff:
- for any distinguished variable X' s.t $X \neq X'$, $s(X) \cap s(X') = \emptyset$ or $s(X) \subset s(X')$,
- $s(X) \cap \bar{Y}_1 = \emptyset$ (the existential variables of $s(X)$ don't appear in the ordinary part),
- the subgraph of G restricted to the variables $X \cup s(X)$ is a tree of root X,
- each node in that tree have no distinguished successor in G.

[7] Note that in our setting, some base atoms may be concept-atoms or role-atoms

For example, $C(X_1) \wedge R_1(X_1, Y_1) \wedge R_2(Y_1, Y_2) \wedge D(Y_2)$ has a tree structure (of root X_1), while neither $C(X_1) \wedge R_1(X_1, Y_1) \wedge R_2(Y_1, X_1)$ nor $R(Y, Y)$ have a tree structure.

A conjunction of concept-atoms and role-atoms which has a tree structure can be expanded by its *Descriptive Support.* The computation of the descriptive support relies on the following definition.

Definition 3.2: *Let ST be a conjunction of concept-atoms and concept-roles having a tree structure of root Z. Let $\bar{Y}$ be its existential variables. The descriptive concept of ST, denoted as $DS[ST]$ is defined as follows:*
- *If $\bar{Y}$ is empty, and therefore ST is of the form $C_1(Z) \wedge \ldots C_n(Z)$ where the C_i's are concept expressions, then: $DS[ST] = C_1 \sqcap \ldots \sqcap C_n$*
- *If $\bar{Y}$ is not empty, and therefore ST is of the form $C_1(Z) \wedge \ldots \wedge C_n(Z) \wedge R_1(Z, {Y_1}^1) \wedge \ldots \wedge R_1(Z, {Y_1}^{n_1}) \wedge \ldots \wedge R_k(Z, {Y_k}^1) \wedge \ldots \wedge R_k(Z, {Y_k}^{n_k}) \wedge Tree({Y_1}^1) \wedge \ldots \wedge Tree({Y_1}^{n_1}) \wedge \ldots \wedge Tree({Y_k}^1) \wedge \ldots \wedge Tree({Y_k}^{n_k})$, where $Tree({Y_i}^j)$ are conjunctions of atoms having a tree structure of root ${Y_i}^j$, then:*

$$\begin{aligned} DS[ST] = C_1 \sqcap \ldots \sqcap C_n \sqcap (\geq n_1\, R_1) \sqcap \ldots \sqcap (\geq n_k\, R_k) \\ \sqcap \forall R_1.(DS[Tree({Y^1}_1)] \sqcap \ldots \sqcap DS[Tree({Y^{n_1}}_1)]) \sqcap \ldots \\ \sqcap \forall R_k.(DS[Tree({Y^1}_k)] \sqcap \ldots \sqcap DS[Tree({Y^{n_k}}_k)]) \end{aligned}$$

If DS is the descriptive concept of a conjunction ST of atoms having a tree structure of root Z, $DS(Z)$ is called the descriptive support of ST.

Example 3.2: Consider the following conjunction of concept-atoms and role-atoms where X_1 is a distinguished variable and Y_1 and Y_2 are existential variables:

$$C(X_1) \wedge R_1(X_1, Y_1) \wedge R_2(Y_1, Y_2) \wedge D(Y_2)$$

It has a tree structure of root X_1. Its descriptive concept is:

$$C \sqcap (\geq 1\, R_1) \sqcap \forall R_1.((\geq 1\, R_2) \sqcap \forall R_2.D))$$

or, rewritten in normal form: $C \sqcap (\geq 1\, R_1) \sqcap \forall R_1.(\geq 1\, R_2) \sqcap \forall R_1.\forall R_2.D$

The descriptive support of $C(X_1) \wedge R_1(X_1, Y_1) \wedge R_2(Y_1, Y_2) \wedge D(Y_2)$ is then:

$$[C \sqcap (\geq 1\, R_1) \sqcap \forall R_1.(\geq 1\, R_2) \sqcap \forall R_1.\forall R_2.D](X_1)$$

The following proposition states that the only way for a conjunction of concept-atoms and role-atoms to be entailed by a concept-atom is to have a tree structure.

Proposition 3.1: *Let $CJ(Z, \bar{Y})$ be a conjunction of concept-atoms and role-atoms. There exists a concept-atom $C(Z)$ such that $C(Z) \models \exists \bar{Y} CJ(Z, \bar{Y})$ iff $CJ(Z, \bar{Y})$ has a tree structure of root Z.*

The full proof is given in [13]. If $CJ(Z, \bar{Y})$ has a tree structure of root Z, it is obvious to show that its descriptive support DS is a concept such that: $DS(Z) \models$

$\exists \bar{Y} CJ(Z, \bar{Y})$. In the case where $CJ(Z, \bar{Y})$ has not a tree structure, for any concept expression C, we can exhibit a model of $C(Z)$ which has a tree structure and which is not a model of $\exists \bar{Y} CJ(Z, \bar{Y})$.

The *compact expansions* are obtained by replacing in the ordinary expansions conjunctions of concept-atoms and role-atoms which have a tree structure by their descriptive supports.

The last step of query expansion consists of expanding each concept-atom and role-atom in every compact or ordinary expansion, as described in the following subsection.

3.3. Expanding concept-atoms and role-atoms

Our algorithm for expanding concept-atoms and role-atoms is inductive. Its termination is guaranteed because it is well-founded on *ground expansions.* Ground expansions of $D(X)$ are those which can be obtained from source atoms that are mapped to possibly complex concept-atoms (of the form $[\forall R_1.\forall R_2.\dots\forall R_k.D](X)$), by conjuncting them with role-atoms involving the same roles. *Complex expansions* are necessary to capture possible rewritings of complex concept-atoms, which cannot be obtained by ground expansions.

Definition 3.3:
Let X be denoting a distinguished variable, and Z be denoting any (distinguished or existential) variable.
- *A concept-atom $C(Z)$ is ground iff:*

- either there exists $v(X) \Rightarrow D(X)$ or $v \sqsubseteq D$ such that C subsumes D: $v(Z)$ is then a ground expansion of $C(Z)$,

- or there exists ground atoms $\forall R.C(U)$ and $R(U, Z)$: a ground expansion of $C(Z)$ is then obtained by conjuncting a ground expansion of $\forall R.C(U)$ and a ground expansion of $R(U, Z)$,

- or, if C is of the form $(\geq n\, R)$, there exists $v(X, Y) \Rightarrow R(X, Y)$ in $\mathcal{I}$: $v(Z, Y_1) \wedge \dots \wedge v(Z, Y_n)$, where the Y_i's are existential fresh variables that appear nowhere else [8], is a ground expansion of $C(Z)$.
- *A role-atom $R(Z, Z)$ is ground iff there exists $v(X, Y) \Rightarrow R(X, Y)$ in $\mathcal{I}$: $v(Z, Z)$ is then a ground expansion of $R(Z, Z)$.*
- *A role-atom $R(Z_1, Z_2)$ is ground iff:*

- either there exists $v(X, Y) \Rightarrow R(X, Y)$ in $\mathcal{I}$: $v(Z_1, Z_2)$ is then a ground expansion of $R(Z_1, Z_2)$,

- or $(\geq n\, R)(Z_1)$ is ground: any ground expansion of $(\geq n\, R)(Z_1)$ is then a ground expansion of $R(Z_1, Z_2)$[9].

Example 3.3: Consider the following implications and terminological constraint establishing a direct mapping between some source atoms and concept and role atoms.

[8] Distinguished or existential variables with different names are considered as distinct variables that cannot be mapped to a same element.

[9] Here, Z_2 is necessarily an existential variable

$v_1(X) \Rightarrow (\forall R_1.\forall R_2.C)(X)$
$v_2(X,Y) \Rightarrow R_1(X,Y)$
$v_3 \sqsubseteq (\geq 2\, R_2)$

$C(Y)$ is a ground concept-atom with an existential variable. It is ground because the atoms $(\forall R_2.C)(X_2)$ and $R_2(X_2,Y)$ are ground. $(\forall R_2.C)(X_2)$ is ground because the atoms $(\forall R_1.\forall R_2.C)(X_1)$ and $R_1(X_1,X_2)$ are ground (they are directly mapped respectively to the source atoms $v_1(X_1)$ and $v_2(X_1,X_2)$). $R_2(X_2,Y)$ is ground because Y is an existential variable and $(\geq 2\, R_2)(X_2)$ is a ground atom (directly mapped to the source atom $v_3(X_2)$). As a result, $C(Y)$ has a single ground expansion: $v_1(X_1) \wedge v_2(X_1,X_2) \wedge v_3(X_2)$. It can be checked that it is a rewriting of $C(Y)$, i.e, $v_1(X_1) \wedge v_2(X_1,X_2) \wedge v_3(X_2) \models \exists Y\, C(Y)$.

It is easy to show that the computation of ground expansions of any ground concept-atom or role-atom terminates and provides rewritings of it. The following proposition states that in the setting of $\mathcal{ALN}$, ground expansions completely characterize possible rewritings for role-atoms and concept-atoms built on simple concept expressions.

Proposition 3.2: *Let cr be a role-atom or a concept-atom $C(Z)$ where C is an atomic concept or the negation of an atomic concept or a number restriction. There exists a rewriting of cr iff cr is a ground atom.*

The full proof is given in [13]. In order to prove that if there exists a rewriting of *cr*, then *cr* is necessarily ground, we consider the initial constraint system corresponding to the existing rewriting, we compute its completion by iteratively applying a set of propagation rules [10]. We then reason by induction on the number of steps that are necessary to obtain the completion from the initial constraint system, considering the different possible forms of *cr*.

In order to get a similar property for complex concept-atoms (i.e atoms of the form $(\forall R.D)(Z)$), we have to consider *complex* expansions for them in addition to ground expansions. Expanding complex concept-atoms is a recursive process implementing the following definition.

Definition 3.4: *Let $(\forall R.D)(Z)$ be a concept-atom such that there exists $(\leq n\, R)(Z)$, $R(Z,U_1) \ldots R(Z,U_n)$ which are ground atoms.*

- Either, $D(U_1),\ldots,D(U_n)$ are ground atoms: complex expansions of $\forall R.D(Z)$ are then obtained by conjuncting the ground expansions of $(\leq n\, R)(Z)$, $R(Z,U_1)\ldots R(Z,U_n)$, $D(U_1),\ldots,D(U_n)$.

- Or, $D(U_1),\ldots,D(U_n)$ are complex concept-atoms that have complex expansions: complex expansions of $\forall R.D(Z)$ are then obtained by conjuncting the ground expansions of $(\leq n\, R)(Z)$, $R(Z,U_1)\ldots R(Z,U_n)$ and the complex expansions of $D(U_1),\ldots,D(U_n)$.

[10] The set of the propagation rules that we consider for $\mathcal{ALN}$ guarantees that there is a unique completion for any initial constraint system built from a set of $\mathcal{ALN}$ expressions.

The following proposition states that the set of terminal expansions of a conjunction of concept-atoms and role-atoms is finite and representative of all the ways of entailing it from terminal atoms.

Proposition 3.3: *Let $Q(\bar{X},\bar{Y})$ be a conjunctive query made of concept-atoms and role-atoms . The set of its terminal expansions is finite and such that:*
(1) For every terminal expansion $E(\bar{X},\bar{Z})$ of $Q(\bar{X},\bar{Y})$: $\exists \bar{Z}\, E(\bar{X},\bar{Z}) \models \exists \bar{Y}\, Q(\bar{X},\bar{Y})$
(2) Let $T(\bar{X},\bar{U})$ be a conjunction of terminal atoms s.t: $\exists \bar{U} T(\bar{X},\bar{U}) \models \exists \bar{Y} Q(\bar{X},\bar{Y})$. The set of atoms appearing in $T(\bar{X},\bar{U})$ includes (up to a renaming of existential variables) all the atoms of atleast one terminal expansion of $Q(\bar{X},\bar{Y})$.

The full proof is given in [13]. It uses the result of Proposition 3.1 and Proposition 3.2. For proving (2), we consider in turn queries containing only ground atoms, then general queries that may contain concept-atoms or role-atoms that are not ground. For each case, we define the notion of depth of query and we prove by induction on the depth of the query that the proposition holds. Each induction step consists of proving that if all the atoms of Q except one appear in $T(\bar{X},\bar{U})$, and if the set of atoms appearing in $T(\bar{X},\bar{U})$ does not include (up to a renaming of existential variables) all the atoms of atleast one expansion of that missing atom, then the so-called *canonical model* of $T(\bar{X},\bar{U})$ is not a model of $Q(\bar{X},\bar{Y})$.

3.4. Completenes and complexity of query processing in PICSEL

Query processing in PICSEL is complete in the sense that the query plans, which are obtained from the query expansion step and which are checked as being valid rewritings by the verification step, completely characterize the set of rewritings of the query. Therefore, the union of the answers resulting from executing those query plans provides the set of the answers that can be obtained from the available sources. The completeness of query processing is a consequence of the completeness of query expansion and of the completeness of the existential entailment used in the verification step. The completeness of existential entailment is proved in [12]. The completeness of query expansion results from the completeness of rule unfolding combined with Proposition 3.3.

In the worst case, query expansion is exponential in terms of the maximum unfolding depth of ordinary atoms [11] and in the maximum size of concept expressions appearing in the query or in the knowledge base. In practice, query expansion has been implemented in Java, and its actual cost appears to be quite reasonable, based on our first experiments. The complete hierarchy of the concepts involved in the domain and source descriptions is computed and stored at compile time. Some indexing is done on the rules too at compile time. As a result, at query time, the expansion of the query can be efficiently obtained from the rules and the concept expressions that are relevant to it. As for the verification step, several optimizations methods are investigated to limit the number of completions that have to be

[11] Let $q(\bar{X})$ an ordinary atom: if q is a base relation, depth $(q(\bar{X})) = 0$; if q is not a base relation, it appears as a consequent of some rules, let $p_1(\bar{X}_1), \ldots, p_n(\bar{X}_n)$ be the atoms appearing in the antecedent of those rules: depth$(q(\bar{X})) = 1 + Max\{depth(p_1(\bar{X}_1)), \ldots, depth(p_1(\bar{X}_n))\}$

constructed in order to simulate case-analysis reasoning. First, we can restrict case-analysis reasoning on the concepts that appear in the query terminal expansions. Second, we are identifying cases for which non costly tests can show that explicit case-analysis on some of those concepts is useless.

4. Conclusions

In the design of the information integration PICSEL system, we have been guided by the theoretical results obtained in [12] and [3], in order to offer a reasonable expressiveness for describing the domain and the sources while maintaining decidability of query answering. The added expressive power of datalog rules and descriptions logics enabled us to model in a natural way a domain and information sources related to tourism. The limitations that we have imposed for computational reasons on the source descriptions have not been an obstacle for a modeling point of view.

PICSEL has similarities with TSIMMIS [7], Information Manifold [11] and Infomaster [10]. Like TSIMMIS, it follows a *Global as view* approach for mapping the source relations with the domain relations. However, in contrast with TSIMMIS, in addition to this simple mapping, it takes into account rather rich constraints about the content of the sources that are expressed as description logics statements. This aspect makes the PICSEL approach a combination of the *Global as view* and *Local as view* approaches. Query processing in Infomaster has similarities with query processing in PICSEL. In particular, an abduction step is used in Infomaster to find the candidate rewritings. However, the language which is used in Infomaster does not contain any description logic component and is less expressive than the one we use in PICSEL for describing both the domain and the content of the sources. Information manifold is based on the use of datalog extended with some features of description logics. The main difference with PICSEL is that it follows a full *Local as view* approach, but a restricted version of CARIN. In PICSEL, we have made the choice to exploit the full expressive power of CARIN, but with a restricted *Local as view* approach. As a matter of fact, it has been shown in [3] that the problem of rewriting queries using views when views and queries are conjonctive queries over $\mathcal{ALN}$ may be undecidable except if some restrictions are imposed on the query and on variables appearing in the views. Some restrictions had been identified in [3] to make the rewriting problem decidable in this setting. The limitation that we impose in PICSEL characterizes a new class of queries and views for which the problem of rewriting queries using views is decidable.

References

[1] Alfred Aho, Yehoshua Sagiv, and Jeffrey D. Ullman. Equivalence of relational expressions. *SIAM Journal of Computing*, (8)2:218–246, 1979.

[2] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6, 1996.

[3] Catriel Beeri, Alon Y.Levy, and Marie-Christine Rousset. Rewriting queries using views in description logics. In *Proceedings of the Sixteenth Symposium on Principles of Database Systems (PODS'97)*, 1997.

[4] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati. Description Logic Framework for Information Integration. In *Proceedings of the Sixth Internation Conference on Principles of Knowledge Representation and Reasoning: KR'98*, 1998.

[5] Diego Calvanese, Giuseppe De Giacomo and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the Seventeenth Symposium on Principles of Database Systems (PODS'98)*, 1998.

[6] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[7] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogenous information sources. In proceedings of IPSJ, Tokyo, Japan, October 1994.

[8] Oliver Duschka and Michael Genesereth. Answering Queries Using Views. In *Proceedings of the Sixteenth Symposium on Principles of Database Systems (PODS'97)*, 1997.

[9] Daniela Florescu, Alon Levy and Alberto Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record Journal*, 27(3), 1998.

[10] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD-97*, 1997.

[11] Alon Y.Levy, Anand Rajamaran, and Joann Ordille. Query-answering algorithms for information agents. In *Proceedings of the thirtenth AAAI conference on Artificial Intelligence: AAAI'96*, 1996.

[12] Alon Levy and Marie-Christine Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence Journal*, 104, pp 165-209, 1998.

[13] Marie-Christine Rousset. Représentation des connaissances et algorithme d'expansion de requêtes dans PICSEL. *Technical report*, in French, available at http://www.lri.fr/ ~ mcr/publis/rapport-picsel.ps, 1998.

[14] Yehoshua Sagiv. Optimizing datalog programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 659–698. Morgan Kaufmann, Los Altos, CA, 1988.

[15] Oded Shmueli. Equivalence of datalog queries is undecidable. *Journal of Logic Programming*, 15:231–241, 1993.

[16] Jeffrey D. Ullman. Information integration using logical views. In *Proceedings of ICDT'97*, 1997.