Query Expansion in Description Logics and Carin

Marie-Christine Rousset
L.R.I. Computer Science Laboratory
C.N.R.S & University of Paris-Sud
Building 490, 91405, Orsay Cedex, France
mcr@lri.fr

Abstract

In this paper, we address the problem of determining the expansions of a query in description logics possibly combined with Datalog rules, and we show its use for information integration. Expanding a query consists of determining all the ways of deriving it from atoms built on some distinguished predicates.

Introduction

Given a knowledge base, expanding a query consists of determining all the ways of deriving it from atoms built on some distinguished predicates. In this paper, we address the problem of determining the expansions of a query in description logics and CARIN. Description Logics are logical formalisms for representing classes of objects (called *concepts*) and their relationships (expressed by binary relations called *roles*). Much of the research in description logics has concentrated on algorithms for checking subsumption between concepts and satisfiability of knowledge bases (see e.g. (Nardi *et al.* 1995) for a survey). When the query is a single concept-atom and the distinguished predicates that are the target of the expansion process are 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. CARIN(Levy & Rousset 1998) combines the expressive powers of function-free Horn rules and description logics by allowing concepts and roles to appear as predicates in the antecedents of the Horn rules.

We show the use of query expansion for information integration and we establish its connection with the problem of rewriting queries using views. 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.

Preliminaries and examples

In the setting of this paper, a CARIN knowledge base Δ contains two components: a set $\Delta_{\mathcal{R}}$ of rules, and a set $\Delta_{\mathcal{T}}$ of concept definitions using the $\mathcal{ALN}$ description logic. $\mathcal{ALN}$ contains the description logics 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 atomic concepts. Concepts and roles are unary and binary predicates that can also appear in the antecedents of the rules. Predicates that do not appear in the description logics component are called *ordinary predicates*. Ordinary predicates areof any arity.

Description logics component in CARIN: Concept definitions are of the form $CN := ConceptExpression$, where CN is a concept name and $ConceptExpression$ is a $\mathcal{ALN}$ concept expression. We assume that a concept name appears in the left hand side of at most one concept definition. A concept name CN depends on a concept name CN' if CN' appears in the definition of CN. We consider only acyclic concept definitions. A set of concept definitions is said *acyclic* if there is no cycle in the concept names dependency relation. An acyclic set of definitions can be unfolded by iteratively substituting every concept name with its definition. 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 of a CARIN knowledge base is a set of rules that are logical sentences of the form: $p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$, where $\bar{X}_1, \ldots, \bar{X}_n, \bar{Y}$ are tuples of variables or constants. We require that the rules are safe, i.e., a variable that appears in $\bar{Y}$ must also appear in $\bar{X}_1 \cup \ldots \cup \bar{X}_n$. The predicates $p_1, \ldots, p_n$ may be either concept names or expressions, role names, or ordinary predicates that do not appear in $\Delta_{\mathcal{T}}$. The predicate q must be an ordinary predicate. We consider non recursive rules. 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.

Semantics of CARIN : The semantics of CARIN knowledge bases 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 predicate of arity n. In particular, it assigns a unary relation C^I to every concept in $\Delta_\mathcal{T}$, and a binary relation R^I over $\mathcal{O}^I \times \mathcal{O}^I$ to every role R in $\Delta_\mathcal{T}$. The extensions of concept and role descriptions are given by the following equations: ($\sharp\{S\}$ denotes the cardinality of a set S):

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

An interpretation I is a model of a knowledge base if it is a model of each of it components. I is a model of $\Delta_\mathcal{T}$ if $CN^I = D^I$ for every concept definition $CN := D$ in $\Delta_\mathcal{T}$. We say that *C is subsumed by D* w.r.t. $\Delta_\mathcal{T}$ if $C^I \subseteq D^I$ in every model I of $\Delta_\mathcal{T}$. An interpretation I is a model of a rule r if, whenever α is a mapping from the variables 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 $\alpha(\bar{Y}) \in q^I$, where $q(\bar{Y})$ is the consequent of r.

Source predicates and atoms: We consider that we are given a set of *source predicates* which represent predicates of special interest. The reason why those source predicates are distinguished depends on varied purposes. In the setting of hypothetical reasoning, source predicates will represent predicates on which assumptions can be made. In a database setting, the source predicates will be EDB predicates representing stored relations, in constast with the IDB predicates whose atoms are necessarily derived from rules and EDB atoms. In a similar way, in an information integration setting, the source predicates will represent the contents of the data sources that have to be integrated.

In the setting of description logics and CARIN, source predicates may be named predicates but also complex concept expressions built over concept names and role names appearing in the knowledge base.

We call a *source atom* an atom $s(\bar{X})$ such that s is a source predicate.

Queries: We consider unions of conjunctive queries of the form: $Q(\bar{X}) : p_1(\bar{X}_1, \bar{Y}_1) \ldots, p_n(\bar{X}_n, \bar{Y}_n)$ where the p_i's are predicates of any arity, some of which may be concept expressions or role names. The variables of $\bar{X} = \bar{X}_1 \cup \ldots \cup \bar{X}_n$ are called the *distinguished* variables of the query: they represent the variables of the query which the user is interested in knowing the instances of, when he asks the query. The variables that are not distinguished (denoted by $\bar{Y}_1 \cup \ldots \cup \bar{Y}_n$) are called *existential* variables of the query: they are existentially quantified in the query 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.

Example 0.1: Suppose that our basic vocabulary contains: (a) the concepts *Flight* denoting a set of flights, *AmCity* and *AmCompany*, which respectively denote the set of American cities and the set of American companies ; (b) the roles *Stop* and *Airline* which respectively denote binary relationships between flights and cities, and flights and airline companies. Suppose that we have the following rules in order to state two different ways for a flight to be convenient for American people:

$$r_1 : Flight(X) \wedge \forall Stop.AmCity(X) \Rightarrow ForAm(X)$$
$$r_2 : Flight(X) \wedge Airline(X, Y) \wedge AmCompany(Y) \Rightarrow ForAm(X)$$

Rule r_1 says that a flight which only stops in American cities is convenient for American people. Rule r_2 says that a flight which has as airline an American company is convenient for American people too.

Suppose that we have the following concept definitions:

$$Flight1 := Flight \sqcap (\leq 1 Stop)$$
$$Flight2 := Flight \sqcap (\geq 1\, Airline) \sqcap \forall Airline.AmCompany$$
$$EastCity := AmCity \sqcap \forall Located.EastCoast$$

*Flight*1 denotes the set of flights that have atmost one stop. *Flight*2 denotes the set of flights which are associated with airlines that are only american companies. Finally, *EastCity* denotes the set of american cities that are located on east coast.

Suppose that we declare as source predicates: the concepts *EastCity*, *Flight*1 and *Flight*2, and the role *Stop*. In an information integration setting, it may express that we have four information sources, one containing American cities from the east coast, a second one containing flights that have at most one stop, a third one containing flights on American airlines, and a last one connecting flights to the cities in which they stop. The query $ForAm(X)$ has two expansions:

1. $Flight1(X) \wedge Stop(X, Y) \wedge EastCity(Y)$,
2. $Flight2(X)$

The first expansion logically entails the query because any instance of X that is a flight which has at most one stop, and which has a stop in a city from the east coast, is necessarily a flight whose all stops are American cities, thus satisfying the antecedent of Rule r_1.

The second expansion logically entails the query because any object that belongs to $Flight2$ is a flight having as airline company an american company, which makes the antecedent of rule r_2 satisfied.

Query Expansion Process

Expanding a query 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 (see section). Then, expansions of (conjunctions of) concept-atoms and role atoms are computed. The process of determining those *terminological expansions* is more complex and is detailed in section .

Ordinary expansions

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 $\Delta_{\mathcal{R}}$. 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 $p(\bar{X})$. If 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 initial atoms (i.e., atoms that are not unifiable with any consequent of rule).

In Example 0.1, the ordinary expansions of the query $ForAm(X)$ are:

1. $Flight(X) \wedge \forall Stop.AmCity(X)$,
2. $Flight(X) \wedge Airline(X, Y) \wedge AmCompany(Y)$.

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 atoms that are made of ordinary predicates, 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 $\bar{Y}_1 \cup \bar{Y}_2$.[1]

The fact that backward-chaining is sound and complete for non recursive Horn clauses without function symbols guarantees that the expansions of the query can be obtained by expanding concept-atoms and role-atoms in every ordinary expansions.

[1]Note that $\bar{X}_1$ and $\bar{X}_2$ (respectively $\bar{Y}_1$ and $\bar{Y}_2$) are not necessarily disjoint

Terminological expansions

Like expansions of ordinary atoms, expanding concept-atoms and role-atoms aims at reaching source atoms from which they can be entailed. However, it raises several issues that did not occur in expanding ordinary atoms. First, from a single but complex concept-atom $C(X)$, we may derive existential sentences made of several concept-atoms and role-atoms in which some variables that do not appear in $C(X)$ are existentially quantified. We then have to consider not only expansions of single atoms but also expansions of existential conjunctive sentences considered as a whole. We show that the existential variables appearing in those existential conjunctive sentences must have a particular binding for those sentences to be entailed by a single concept-atom. We provide an algorithm to transform those sentences into single concept-atoms by computing their *descriptive support*. Second, in contrast with ordinary atoms, the entailment of concept-atoms and role-atoms is not supported by explicit dependency links expressed by rules in the knowledge base. We define inductively the different ways of deriving a concept-atom (respectively a role-atom) in $\mathcal{ALN}$. For our inductive definition to be well-founded, we identify *ground* concept-atoms and role-atoms as intermediate atoms between source atoms and query atoms. As a result, we obtain an expansion algorithm that provides *candidate expansions*. The subtle point is that case analysis reasoning may be necessary to verify that those candidate expansions do actually entail the query.

Descriptive Support: Definition and Algorithm. We say that a variable Z' is a direct successor of a variable 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, and let $CJ(X)$ the conjunctive sentence contained in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$ composed of concept-atoms involving X and all the 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.

Definition 0.1: *A conjunctive sentence $CJ(X)$ contained in $CJ_{term}(\bar{X}_2, \bar{Y}_2)$ has a* tree structure *iff:*
- for every distinguished variable X' such that $X \neq X'$, $s(X) \cap s(X') = \emptyset$,
- the subgraph of G restricted to the variables $X \cup s(X)$ is a tree of root X.

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 replaced by a single concept-atom, using its *Descriptive Support*. The descriptive support is a concept expression which is obtained by the algorithm described in Figure 1.

procedure Descriptive_Support (ST(Z))
`/*` $ST(Z)$ `is a conjunction of concept-atoms and concept-roles having a tree structure of root` Z `and of existential variables` $\bar{Y}$`*/`
`/* The algorithm returns a concept` DS `and a concept atom` $DS(Z)$ `*/`

If $ST(Z)$ `is of the form:` $C_1(Z) \wedge \ldots C_n(Z)$,
`where the` C_i`'s are concept expressions`
Then: $DS = C_1 \sqcap \ldots \sqcap C_n$
Else If $ST(Z)$ `is of the form:`
$C_1(Z) \wedge \ldots \wedge C_n(Z) \wedge R_1(Z, {Y_1}^1) \wedge \ldots \wedge R_1(Z, {Y_1}^{n_1})$
$\wedge \ldots \wedge R_k(Z, {Y_k}^1) \wedge \ldots \wedge R_k(Z, {Y_k}^{n_k})$
$\wedge Tree({Y_1}^1) \wedge \ldots Tree({Y_1}^{n_1})$
$\wedge \ldots \wedge Tree({Y_k}^1) \wedge \ldots Tree({Y_k}^{n_k})$,
`where` $Tree({Y_i}^j)$`'s are conjunctions of atoms having a tree structure of root` Y_i^j
Then:
for $i \in [1..k]$ **do:** **for** $j \in [1..n_i]$ **do:**
$DS_i^j := Descriptive\_Support(Tree(Y_i^j))$
$DS := C_1 \sqcap \ldots \sqcap C_n \sqcap (\geq n_1 R_1) \sqcap \ldots (\geq n_k R_k)$
$\sqcap \forall R_1.({DS^1}_1 \sqcap \ldots \sqcap {DS^{n_1}}_1) \sqcap \ldots$
$\sqcap \forall R_k.({DS^1}_k \sqcap \ldots \sqcap {DS^{n_k}}_k)$

return DS `as descriptive support of` $ST(Z)$
return $DS(Z)$ `as DS-atom of` $ST(Z)$

Figure 1: Descriptive Support algorithm

Example 0.2 : 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 support returned by the procedure of Figure 1 is the concept:

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

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 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 0.1 : *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.*

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 $\exists \bar{Y} CJ(Z, \bar{Y})$.

Determining Candidate Expansions: We first define ground concepts and roles (respectively, concept-atoms and role-atoms). The intuition is that ground concepts and roles result direcly or indirectly of the unfolding of complex source concepts (i.e., of the form $\forall R_1.\forall R_2.\ldots.\forall R_k.D$). Formally, they are defined inductively as follows.

Definition 0.2:

- *A concept C is ground iff:*

- either, there exists a source concept D which is subsumed by C,

- or, C is of the form $(\geq n R)$ and R is a source role,

- or, $\forall R.C$ and R are ground

- *A role R is ground iff either, R is a source role, or, $(\geq n R)$ is ground.*

- *A concept-atom $C(Z)$ (respectively a role-atom $R(Z, Z'))$ [2] is ground iff C (respectively R) is ground.*

In Example 0.1, *AmCompany* is ground because $\forall Airline.AmCompany$ and $Airline$ are ground. $\forall Airline.AmCompany$ is ground because it subsumes the source concept $Flight2$. $Airline$ is ground because $(\geq 1\, Airline)$ is ground, since $(\geq 1\, Airline)$ subsumes the source concept $Flight2$.

The process of expanding a conjunction of concept-atoms and role-atoms consists then of:

1. replacing any set of conjuncts that has a tree structure by its DS-atom, if it is ground,
2. expanding in the resulting conjunction, any concept-atom and role-atom, as described in the algorithm of figure 2. As a result, we obtain a set of *terminal expansions* of the form $s_1(\bar{X_1}, \bar{Y_1}) \wedge \ldots \wedge s_k(\bar{X_k}, \bar{Y_k}) \wedge remainder(\bar{X}, \bar{Y})$, where $remainder(\bar{X}, \bar{Y})$ is a conjunction of atoms that have no expansion.

The following proposition states that it provides all the ways of deriving a conjunction of concept-atoms and of role-atoms from atoms that are either source atoms or atoms with no expansion.

Proposition 0.2: *Let $\exists \bar{Y}\, Q(\bar{X}, \bar{Y})$ be a conjunction of concept-atoms and of role-atoms . Let $CJ_S(\bar{X}, \bar{Z})$ be a conjunction of atoms that are either source atoms or atoms that have no expansion.*
$\exists \bar{Z}\, CJ_S(\bar{X}, \bar{Z}) \models \exists \bar{Y} Q(\bar{X}, \bar{Y})$ iff $CJ_S(\bar{X}, \bar{Z})$ contains (up to renaming existential variables) a terminal expansion of $Q(\bar{X}, \bar{Y})$.

[2] The variables Z and Z' may be existentially quantified.

```
procedure expand(cr)
/* cr is a concept-atom or a role-atom */

/* the algorithm returns a set of
conjunctions of atoms that are either source
atoms or atoms that have no expansion*/

result := ∅
If cr is a source atom Then:
   result := result ∪ {cr}
If cr is a ground atom Then:
   If cr is a concept-atom C(Z) such that
      C subsumes a source concept D
   Then:  result := result ∪ {D(Z)}
   If cr is a concept-atom C(Z) such that
      there exists ground atoms of the form
      ∀R.C(U) and R(U,Z)
   Then:
      E1 := expand(∀R.C(U))
      E2 := expand(R(U,Z))
      for every cj ∈ E1 do:
         for every cj' ∈ E2 do:
            result := result ∪ {cj ∧ cj'}
   If cr is a concept-atom (≥ n R)(Z) such that
      R is a source role
   Then:  result := result ∪ {R(Z,Y_1) ∧ ... ∧
R(Z,Y_n)},
      where Y_1,...,Y_n are fresh variables
If cr is of the form ∀R.C(X) Then :
   If there exists ground atoms
      (≤ n R)(X), R(X,Z_1),...,R(X,Z_n)
   Then:
      E_≤  := expand ((≤ n R)(X))
      E_R1 := expand (R(X,Z_1))
      ...
      E_Rn := expand (R(X,Z_n))
      E_C1 := expand (C(Z_1))
      ...
      E_Cn := expand (C(Z_n))
      for every cj ∈ E_≤ do:
         for every cj_1 ∈ E_R1,...,cj_n ∈ E_Rn do:
            for every cj'_1 ∈ E_C1,...,cj'_n ∈ E'_Cn do:
               result := result∪
                  {cj ∧ cj_1 ∧ ... ∧ cj_n ∧ cj'_1 ... ∧ cj'_n}
If result = ∅ Then:
   mark cr as having no expansion
   result := {cr}
return:  result
```

Figure 2: Algorithm computing terminal expansions

For proving Proposition 0.2, 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 $CJ_S(\bar{X}, \bar{Z})$ contains all the atoms of Q except one and if it does not contain any expansion of that atom, then it does not entail the query. For doing so, we exhibit a *canonical model* of $CJ_S(\bar{X}, \bar{Z})$ and we show that it is not a model of $Q(\bar{X}, \bar{Y})$.

The set of *candidate expansions* are obtained by keeping only source atoms from every terminal expansion.

An immediate corollary of Proposition 0.2 is that the query cannot be entailed by a conjunction of source atoms which is not contained in a candidate expansion. However, the converse is not necessarily true: we have to check whether, possibly by case-analysis reasoning, the candidate expansions do entail the query.

Verification of the expansions: Let $s_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge s_i(\bar{X}_i, \bar{Y}_i)$ be a candidate expansion. It comes from a set of terminal expansions of the query of the form: $s_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge s_i(\bar{X}_i, \bar{Y}_i) \wedge remainder(\bar{X}, \bar{Y})$, where $remainder(\bar{X}, \bar{Y})$ is a conjunction of atoms that have no expansion.

We use the *existential entailment* algorithm, which is described in (Levy & Rousset 1998), 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 $s_1(\bar{X}_1, \bar{Y}_1) \wedge \ldots \wedge s_i(\bar{X}_i, \bar{Y}_i)$, 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.

In Example 0.1, the terminal expansions contain only source atoms. Therefore, the candidate expansions are directly obtained and do not not have to be checked. The following example illustrates a case where candidate expansions are not reduced to terminal expansions and then query entailment has to be checked.

Example 0.3: Let consider the query $Q(X)$ defined by the two rules:

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

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

Consider that we have only one source predicate:

$Flight3 : Flight \sqcap \forall Stop.AmCity$

The ordinary expansions of $Q(X)$ are:

$Flight(X) \wedge (\leq 1\, Stop)(X)$

$Flight(X) \wedge Stop(X, Y) \wedge AmCity(Y)$, which can be replaced by its DS-atom:

$$Flight(X) \wedge (\geq 1\, Stop)(X) \wedge \forall Stop.AmCity(X)$$

The terminal expansions of $Q(X)$ are:

$Exp_1 : Flight3(X) \wedge (\leq 1\, Stop)(X)$

$Exp_2 : Flight3(X) \wedge (\geq 1\, Stop)(X)$

They both contain 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 expansion $Flight3(X)$. Existential entailment is then used to check whether $Flight3(X) \models Exp_1 \vee Exp_2$. 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, $Flight3(X)$ is a candidate expansion which is validated as an actual expansion of $Q(X)$.

The previous results can be summarized by the following theorem stating that query expansion is sound and complete for $\mathcal{ALN}$-CARIN.

Theorem 0.1: *Let $\Delta_{\mathcal{R}} \cup \Delta_{\mathcal{T}}$ be a $\mathcal{ALN}$-CARIN knowledge base, and let $\mathcal{S}$ be a set of source predicates. Let $Q(\bar{X})$ a conjunctive query. Let $Exp_1(\bar{X}), \ldots, Exp_n(\bar{X})$ be the set of candidate expansions of $Q(\bar{X})$ that have been validated.*

- *$\forall i \in [1..n], Exp_i(\bar{X}) \cup \Delta_{\mathcal{R}} \cup \Delta_{\mathcal{T}}$ is satisfiable and entails $Q(\bar{X})$*
- *Any conjunction of source atoms that entails $Q(\bar{X})$ contains (up to renaming existential variables) one of those expansions.*

This is the result of (1) the soudness and completeness of backward-chaining for non recursive free-function Horn rules, and (2) Proposition 0.1 and 0.2.

Application to Information Integration

Query expansion in $\mathcal{ALN}$-CARIN is the core algorithmic tool for building query plans in PICSEL (Lattes & Rousset 1998). PICSEL is an information integration system over sources that are distributed and possibly heterogeneous. A first feature that distinguishes PICSEL from existing integration information systems like Information Manifold (Y.Levy, Rajamaran, & Ordille 1996), SIMS (Arens, Knoblock, & Shen 1996), or Infomaster (Genesereth, Keller, & Duschka 1997) is its full use of the combination of Datalog with description logics to represent both the domain of application and the contents of information sources. Note that Information Manifold already used Datalog extended with some features of description logics and thus was based on a first and restricted version of CARIN. In PICSEL, the contents of the sources are described by single atoms, while they can be arbitrary queries (called *views*) in Information Manifold and Infomaster. As a result, answering queries in Information Manifold and Infomaster is based on rewriting queries using views. The problem of rewriting queries using views has been studied for several classes of relational queries ((Ullman 1997)). In (Beeri, Y.Levy, & Rousset 1997), 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 variables appearing in the views. The problem of expanding a query is a particular case of the problem of rewriting queries using views, in which views are reduced to single atoms. When the queries and the views are expressed using description logics, this limitation guarantees that query rewriting is decidable. It is important to note that allowing complex concept expressions as predicates makes that this limitation that is imposed for computational reasons is not a so big limitation from an expressive power point of view.

Conclusion

We have provided a sound and complete algorithm for query expansion in $\mathcal{ALN}$-CARIN. It can be used for backward-chaining reasoning on $\mathcal{ALN}$ knowledge bases. For knowledge bases of big size, it provides a query evaluation algorithm which may be more efficient than the algorithms based on constraint systems which saturate the knowledge bases by applying propagation rules for building completions (Nardi *et al.* 1995).

References

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

Beeri, C.; Y.Levy, A.; and Rousset, M.-C. 1997. Rewriting queries using views in description logics. In *Proceedings of the Sixteenth Symposium on Principles of Database Systems (PODS'97).*

Genesereth, M. R.; Keller, A. M.; and Duschka, O. M. 1997. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD-97.*

Lattes, V., and Rousset, M.-C. 1998. The use of carin language and algorithms for information integration: The picsel project. In *Proceedings of the workshop on Intelligent Information Integration associated with the European Conference on Artificial Intelligence ECAI'98, Brighton, UK.*

Levy, A., and Rousset, M.-C. 1998. Combining horn rules and description logics in carin. *Artificial Intelligence* 101.

Nardi, D.; Donini, F.; Lenzerini, M.; and Schaerf, A. 1995. Reasoning in description logics. In *Principles of Artificial Intelligence.* Springer-Verlag.

Ullman, J. D. 1997. Information integration using logical views. In *Proceedings of the Int. Conf. on Database Theory, Athens, Greece.*

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