Backward Reasoning in Aboxes for Query Answering

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 answering (unions of) conjunctive queries on Aboxes. We propose a backward-chaining query evaluation based on query expansion. Expanding a query consists of determining all the ways of deriving it from atoms built on some distinguished predicates.

1 Introduction

Description Logics (DL) are logical formalisms for representing and reasoning on complex classes of individuals (called *concepts*) and their relationships (expressed by binary relations called *roles*). A DL knowledge base has two components: (1) the Tbox contains a set of concept descriptions and represents the general schema modeling the domain of interest ; (2) the Abox is a partial instantiation of this schema made of a set of assertions relating either individuals to classes, or individuals to each others.

It has been pointed out that DL can be useful in a database setting for several purposes and in particular, for data modeling (see [2] for a survey) and for information integration ([1; 3]). An interesting characteristic of DL systems for databases is that they allow a natural representation of various forms of incomplete information. In a Abox, we do not have to provide *all* the information about an individual. For example, it is possible to assert $(\geq\ 2\,child)(Bob)$ denoting that Bob has atleast two children, but without specifying *who* the children are. The system will still be able to infer that Bob is a parent from this partial information. Another advantage of DL systems is that they provide inference services, based on the formal semantics of the underlying description logic. In particular, given a set of concept definitions, the resulting subsumption hierarchy can be automatically computed. Another important inference service is automatic instance recognition: given the information available about an individual in the Abox, and given the concept definitions contained in the TBox, a DL system will automatically place the individual in the extension of the concepts it belongs to. Those inference services are based on algorithms for checking subsumption between concepts and satisfiability of knowledge bases (see e.g. [6] for a survey). However, those algorithms are not sufficient for fully reasoning on Aboxes. In particular, the problem of answering (unions of) conjunctive queries on Aboxes is a core algorithmic issue in order to export the DL services to the database world. In addition, in several domains of applications, it would be very useful to manage large and persistent Aboxes. In such a setting, a standard algorithmic approach based on building completions ([6]) is not feasible.

In this paper, we propose a backward-chaining query evaluation based on query expansion. We provide a sound and complete algorithm for query expansion in the $\mathcal{ALN}$ description logic. As a result, we obtain a new class of queries and views for which the problem of rewriting queries using views is decidable.

2 Preliminaries and examples

$\mathcal{ALN}$ Tboxes and Aboxes: $\mathcal{ALN}$ 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 atomic concepts. We consider acyclic Tboxes made of concept definitions of the form $CN := \mathit{ConceptExpression}$, where CN is a concept name and $\mathit{ConceptExpression}$ is a $\mathcal{ALN}$ concept expression. We consider that every concept has been unfolded (by iteratively substituting every concept name with its definition) 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.

An Abox is a set of *concept-facts* and *role-facts*. A concept-fact is of the form $C(o)$ where o is an individual and C is a $\mathcal{ALN}$ concept expression. A role-fact is of the form $R(o_1, o_2)$ where o_1 and o_2 are individuals and R is a role name.

The semantics of $\mathcal{ALN}$ 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 $\alpha^I(a) \in \mathcal{O}^I$, a unary relation C^I to every concept, and a binary relation R^I over $\mathcal{O}^I \times \mathcal{O}^I$ to every role R. The extensions of concept descriptions are given by the following equations:

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

We consider a database setting, in which the concepts defined in the Tbox and the role names appearing in the concept definitions are EDB predicates. We are interested in querying a persistent Abox composed of stored extensions of the EDB predicates. The EDB predicates may be complex concept expressions built over concept names and role names. In this respect, they can be seen as (materialized) views over those basic concept and role names.

Queries: We consider unions of conjunctive queries of the form: $Q(\bar{X}) : \exists \bar{Y}_1 \ldots \bar{Y}_n\, p_1(\bar{X}_1, \bar{Y}_1) \ldots, p_n(\bar{X}_n, \bar{Y}_n)$ where the $p_i(\bar{X}_i, \bar{Y}_i)$'s are concept-atoms or role-atoms. We call *concept-atom* an atom $p(X)$ where p is a $\mathcal{ALN}$ concept expression, and *role-atom* an atom $r(X,Y)$ where r is a role name. 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.

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

In contrast with conventional relational databases, some inference may be needed to answer queries on Aboxes. In particular, the query has to to be rewritten in terms of EDB predicates which are the only ones that are stored. The rewriting of the query is obtained by computing the set of its *expansions*. Expanding a query consists of determining all the ways of logically deriving it from atoms built on some distinguished predicates. In the setting of this paper, the distinguished predicates that are the target of the expansion process are the EDB predicates which are defined in the Tbox whose Abox is an instantiation.

Example 2.1: Suppose that we have a persistent Abox made of extensions of EDB predicates corresponding to the roles *Stop* and *Airline* and the concepts *Flight*1, *Flight*2 and *EastCity* described as follows:

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

*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 have the following query stating two different ways for a flight to be considered as convenient for American people:

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

The first conjunction says that a flight which only stops in American cities is convenient for American people. The second conjunction says that a flight which has as airline an American company is convenient for American people too.

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, and thus satisfies the first conjunction of the query.

The second expansion logically entails the query because any object that belongs to *Flight*2 is a flight having as airline company an american company, which makes the second conjunction of the query satisfied.

These two expansions provide queries that can be directly posed to the stored Abox, seen as a conventional relational database, by using a standard SQL-like query language and evaluation. The fact that they both logically entail the initial query guarantees that the answers that they provide are also answers to that initial query. The subtle point is to be sure that those answers are all the possible answers that can be obtained for the initial query. This is guaranteed by the completeness of the query expansion process for $\mathcal{ALN}$, which is described in the next section.

3 Query Expansion Process

Expanding a query is an iterative process based on successive steps of expansions that mirror backward reasoning. However, it raises several issues that do not occur in backward reasoning with 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, 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 EDB 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. Let $CJ(\bar{X},\bar{Y})$ be a conjunction of concept-atoms and of role-atoms, whose $\bar{X}$ are the distinguished variables and $\bar{Y}$ are the existential variables. We say that a variable Z' is a direct successor of a variable Z in $CJ(\bar{X},\bar{Y})$ if there exists an atom $R(Z,Z')$ in $CJ(\bar{X},\bar{Y})$. The successor relationship is the transitive closure of the direct successor relationship. For every $X \in \bar{X}$, let $s(X)$ be the set of all the successors of X that are existential variables, and let $CJ(X)$ the conjunctive sentence made of atoms of $CJ(\bar{X},\bar{Y})$ and 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 $CJ(\bar{X},\bar{Y})$ 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 a conjunctive sentence $CJ(X)$ contained in $CJ(\bar{X},\bar{Y})$ has a *tree structure* iff:
- for every distinguished variable X' such that $X \neq X'$, $s(X) \cap s(X') = \emptyset$ or $s(X) \subset s(X')$,
- the subgraph of G restricted to the variables $X \cup s(X)$ is a tree of root X,
- each node in that tree has no distinguished successor in G.

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 3.1 : 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 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.*

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 EDB 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 3.1:

- *A concept C is ground iff:*

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

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

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

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

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

In Example 2.1, *AmCompany* is ground because $\forall$*Airline.AmCompany* and *Airline* are ground. $\forall$*Airline.AmCompany* is ground because it subsumes the EDB concept *Flight*2. *Airline* is ground because $(\geq 1\,Airline)$ is ground, since $(\geq 1\,Airline)$ subsumes the EDB concept *Flight*2.

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 EDB atoms or atoms with no expansion.

Proposition 3.2: *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 EDB 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 $Q(\bar{X},\bar{Y})$.

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

procedure expand(cr)
/* cr is a concept-atom or a role-atom */
/* the algorithm returns a set of conjunctions of atoms that are either EDB atoms or atoms that have no expansion*/
$result := \emptyset$
If cr is a EDB atom **Then:**
  $result := result \cup \{cr\}$
If cr is a ground atom **Then:**
  If cr is a concept-atom $C(Z)$ such that
    C subsumes a EDB concept D
  Then: $result := result \cup \{D(Z)\}$
  If cr is a concept-atom $C(Z)$ such that
    there exists ground atoms of the form
    $\forall R.C(U)$ and $R(U,Z)$
  Then:
    E1 := expand($\forall R.C(U)$)
    E2 := expand($R(U,Z)$)
    for every $cj \in E1$ **do:**
      for every $cj' \in E2$ **do:**
        $result := result \cup \{cj \wedge cj'\}$
  If cr is a concept-atom $(\geq n\,R)(Z)$ such that
    R is a EDB relation
  Then: $result := result \cup \{R(Z,Y_1)\wedge\ldots\wedge R(Z,Y_n)\}$,
    where $Y_1,\ldots,Y_n$ are fresh variables
If cr is of the form $\forall R.C(X)$ **Then :**
  If there exists ground atoms
    $(\leq n\,R)(X), R(X,Z_1),\ldots,R(X,Z_n)$
  Then:
    $E_{\leq}$:= expand $((\leq n\,R)(X))$
    E_{R_1} := expand $(R(X,Z_1))$
    ...
    E_{R_n} := expand $(R(X,Z_n))$
    E_{C_1} := expand $(C(Z_1))$
    ...
    E_{C_n} := expand $(C(Z_n))$
    for every $cj \in E_{\leq}$ **do:**
      for every $cj_1 \in E_{R_1},\ldots,cj_n \in E_{R_n}$ **do:**
        for every $cj'_1 \in E_{C_1},\ldots,cj'_n \in E'_{C_n}$ **do:**
          $result := result \cup$
            $\{cj \wedge cj_1 \wedge \ldots \wedge cj_n \wedge cj'_1 \ldots \wedge cj'_n\}$
If $result = \emptyset$ **Then:**
  mark cr as having no expansion
  $result := \{cr\}$
return: $result$

Figure 2: Algorithm computing terminal expansions

An immediate corollary of Proposition 3.2 is that the query cannot be entailed by a conjunction of EDB 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 Candidate 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 [5], 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. It is important to note that case analysis reasoning is guided by the remainders, thus restricting the number of branchings in the building of completions. The previous results can be summarized by the following theorem stating that query expansion is sound and complete for $\mathcal{ALN}$.

Theorem 3.1:

Let $\mathcal{A}$ be an Abox made of the extensions of a set of given $\mathcal{ALN}$ concept and role expressions. Let $Q(\bar{X})$ be a conjunctive query made of a conjunction of $\mathcal{ALN}$ concept-atoms and role-atoms. Let $Exp_1(\bar{X}), \ldots, Exp_n(\bar{X})$ be the set of candidate expansions of $Q(\bar{X})$ which have been verified:

- $\forall i \in [1..n], Exp_i(\bar{X}) \models Q(\bar{X})$

- *Any conjunction of EDB atoms that entails $Q(\bar{X})$ contains (up to renaming existential variables) one of those expansions.*

4 Conclusion

The expansion process that we have described in this paper provides an algorithm for evaluating union of conjunctive queries against (persistent) $\mathcal{ALN}$ Aboxes. This work has been easily extended to the setting of Carin ([5]), by combining the expansion process with a standard backward-chaining algorithm for non recursive function-free Horn rules.

As a result, we obtain a new class of queries and views for which the problem of rewriting queries using views is decidable. The problem of rewriting queries using views has been studied for several classes of relational queries ([7]). From a relational point of view, a persistent Abox is a set of (materialized) views which are role-atoms and concept-atoms possibly built on complex $\mathcal{ALN}$ concept expressions. In [1], 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 queries and on the variables appearing in the views. In this paper, we have shown that the expansion process computes a representative set of all the possible rewritings of queries in terms of views, where queries are union of conjunctive queries made of $\mathcal{ALN}$ concept-atoms and role-atoms, and views are reduced to *single* (concept or role) atoms. This limitation guarantees that query rewriting is decidable. It is important to note that allowing complex concept expressions in those single atoms makes that this limitation that is imposed for computational reasons is not a so big limitation from an expressive power point of view.

Query expansion in $\mathcal{ALN}$-Carin is the core algorithmic tool for building query plans in the information integration Picsel [4] system.

References

[1] 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.

[2] Alex Borgida. Description logics in data management. *IEEE Trans. on Know. and Data Engineering*,7(5), 1995.

[3] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati. Description Logic Framework for Information Integration. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.

[4] Veronique Lattes and Marie-Christine Rousset. The use of Carin language and algorithms for information integration: The Picsel project. In *Proceedings of the ECAI'98 workshop on Intelligent Information Integration, Brighton, UK*, 1998.

[5] Alon Levy and Marie-Christine Rousset. Combining horn rules and description logics in Carin. *Artificial Intelligence*, 101, 1998.

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

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