Concept Languages as Query Languages

Maurizio Lenzerini, Andrea Schaerf

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Salaria 113, 00198 Roma, Italia

Abstract

We study concept languages (also called terminological languages) as means for both defining a knowledge base and expressing queries. In particular, we investigate on the possibility of using two different concept languages, one for asserting facts about individual objects, and the other for querying a set of such assertions. Contrary to many negative results on the complexity of terminological reasoning, our work shows that, provided that a limited language is used for the assertions, it is possible to employ a richer query language while keeping the reasoning process tractable. We also show that, on the other hand, there are constructs that make query answering inherently intractable.

1 Introduction

Concept languages (CLs, also called terminological languages) provide a means for expressing knowledge about concepts, i.e. classes of individuals with common properties.

A concept is built up of two kinds of symbols, primitive concepts and primitive roles. These primitives can be combined by various language constructs yielding complex concepts. Different languages are distinguished by the constructs they provide. Concept languages are given a Tarski-style semantics: an interpretation interprets concepts as subsets of a domain and roles as binary relations over the domain.

Much of the research on terminological reasoning (see [2,3,5,9,10]) aims at characterizing CLs with respect to both expressive power and computational complexity of computing subsumption, i.e. checking if one concept is always a superset of another.

Other recent work (see [4,6]) deals with the problem of using a CL for building what we call a concept-based knowledge base, i.e. a set of assertions about the membership relation between individuals and concepts, and between pairs of individuals and roles.

It is interesting to observe that little attention has been paid to studying CLs as query languages, i.e. as means for extracting information from a concept-based knowledge base. In the present paper we deal with this problem, with the main goal of identifying an optimal compromise between expressive power and computational tractability for both the assertional and the query language. Our work has been carried out with the following underlying assumptions:

- The assertional language is at least as powerful as $\mathcal{FL}^-$ [2], which is generally considered as the minimal concept language.
- A query is formulated in terms of a concept C, with the meaning of asking for the set of all the individuals x such that the knowledge base logically implies that x is an instance of C.
- Since we want to be able to extract from the knowledge base at least the stored information, the query language is at least as expressive as the assertional language.
- The computational complexity of query answering is measured with respect to the size of both the knowledge base and the concept representing the query.

The main result of this paper is to show that one can use a rich CL for query formulation without falling into the computational cliff, provided that a tractable language is used for constructing the knowledge base.

It is worth mentioning that the idea of using a query language richer than the assertional language is not new. For example, relational data bases, which are built up by means of a very limited data definition language, are queried using a full first order language, called relational calculus. Another example is the work by Levesque in [8], where a first order knowledge base

This work was partly funded by ESPRIT BRA 3012 (Compulog) and by the Italian CNR, under Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Linea di ricerca IBRIDI.

is queried by means of a richer language including a modal operator.

In order to apply this idea in the context of concept-based knowledge bases, we make use of $\mathcal{AL}$ [10] as assertional language, and we define a suitable new language $\mathcal{QL}$ for query formulation. $\mathcal{AL}$ is a tractable extension of $\mathcal{FL}^-$ with a constructor for denoting the complement of primitive concepts, whereas $\mathcal{QL}$ is an extension of $\mathcal{AL}$ to express qualified existential quantification on roles, role conjunction, and collection of individuals.

Another result of our work is that $\mathcal{AL}$ and $\mathcal{QL}$ are almost maximally expressive with respect to the tractability of query answering. In particular, by analyzing the constructs usually considered in terminological languages, we show that, if one aims at retaining tractability, there are inherent limits to the expressive power of both the assertional and the query language.

The paper is organized as follows. In Section 2 we provide some preliminaries on CLs. In Section 3, we deal with the problem of checking subsumption between a concept of $\mathcal{QL}$ and a concept of $\mathcal{AL}$. In Section 4, we make use of the results of Section 3 for devising a polynomial method for answering queries to an $\mathcal{AL}$-knowledge base using $\mathcal{QL}$ as query language. In Section 5 we discuss the limits for the tractability of query answering. Finally, conclusions are drawn in Section 6.

For the sake of brevity, most of the proofs are omitted. They can be found in [7].

2 Preliminaries

In this paper, we consider a family of *concept languages* whose general description can be found in [5,10].

We are particularly interested in the language $\mathcal{AL}$, where *concepts* (denoted by the letters C and D) are built out of *primitive concepts* (denoted by the letter A) and *primitive roles* according to the syntax rule

$$C, D \longrightarrow A \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R$$

where R denotes a *role*, that in $\mathcal{AL}$ is always primitive (other languages provide constructors for roles too).

Both $\mathcal{FL}^-$ and $\mathcal{AL}$ provide a restricted form of existential quantification, called unqualified: the construct $\exists R$ denotes the set of objects a such that there exists an object b related to a by means of the role R: the existential quantification is unqualified in the sense that no condition can be stated on b other than its existence.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^\mathcal{I}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept to a subset of $\Delta^\mathcal{I}$ and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that the following equations are satisfied:

$$(\neg A)^\mathcal{I} = \Delta^\mathcal{I} \setminus A^\mathcal{I}, \quad (C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I},$$
$$(\forall R.C)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \forall (a,b) \in R^\mathcal{I}. b \in C^\mathcal{I}\},$$
$$(\exists R)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \exists (a,b) \in R^\mathcal{I}\}.$$

An interpretation $\mathcal{I}$ is a *model* for a concept C if $C^\mathcal{I}$ is nonempty. A concept is *satisfiable* if it has a model and *unsatisfiable* otherwise. We say that C is *subsumed* by D if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for every interpretation $\mathcal{I}$, and C is *equivalent* to D if C and D are subsumed by each other.

More general languages are obtained by adding to $\mathcal{AL}$ the following constructs:

- qualified existential quantification, written as $\exists R.C$, and defined by $(\exists R.C)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \exists (a,b) \in R^\mathcal{I}. b \in C^\mathcal{I}\}$. The difference with unqualified existential quantification is that in this case a condition is specified on object b, namely that it must be an instance of the concept C;
- disjunction of concepts, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$;
- intersection of roles, $(Q \sqcap R)^\mathcal{I} = Q^\mathcal{I} \cap R^\mathcal{I}$;
- collection of individuals (see [1]), written as $\{a_1, \ldots, a_n\}$, where each a_i is a symbol belonging to a given alphabet O. In order to assign a meaning to such a concept, the interpretation function $\cdot^\mathcal{I}$ has to be extended by injectively associating an element of $\Delta^\mathcal{I}$ with each symbol in O. The semantics of $\{a_1, \ldots, a_n\}$ is then defined by $\{a_1, \ldots, a_n\}^\mathcal{I} = \{a_1^\mathcal{I}, \ldots, a_n^\mathcal{I}\}$.

In [5,10] a calculus for checking concept satisfiability is presented. The calculus operates on constraints of the forms $x\colon C$, xRy, where x, y are variables belonging to an alphabet V, C is a concept and R is a role.

Let $\mathcal{I}$ be an interpretation. An *$\mathcal{I}$-assignment* α is a function that maps every variable to an element of $\Delta^\mathcal{I}$; α *satisfies* $x\colon C$ if $\alpha(x) \in C^\mathcal{I}$, and α *satisfies* xRy if $(\alpha(x), \alpha(y)) \in R^\mathcal{I}$. A *constraint system* (i.e. a finite, nonempty set of constraints) S is *satisfiable* if there is an interpretation $\mathcal{I}$ and an $\mathcal{I}$-assignment α such that α satisfies every constraint in S.

It is easy to see that a concept C is satisfiable iff the constraint system $\{x\colon C\}$ is satisfiable. In order to check C for satisfiability, the calculus starts with the constraint system $S = \{x\colon C\}$, adding constraints to S until either a contradiction is generated or an interpretation satisfying C can be obtained from the resulting system. Constraints are added on the basis of a suitable set of so-called *propagation rules*, whose form depends on the constructs of the language. The propagation rules for the language $\mathcal{AL}$ are:

1. $S \rightarrow_\sqcap \{x\colon C_1,\ x\colon C_2\} \cup S$
   if $x\colon C_1 \sqcap C_2$ is in S,
   and $x\colon C_1$ and $x\colon C_2$ are not both in S

2. $S \rightarrow_{\forall} \{y{:}C\} \cup S$
   if $x{:}\forall P.C$ is in S,
   xPy is in S, and $y{:}C$ is not in S

3. $S \rightarrow_{T\exists} \{xPy\} \cup S$
   if $x{:}\exists P$ is in S, y is a new variable
   and there is no z such that xPz is in S

4. $S \rightarrow_{\perp} \{x{:}\perp\}$
   if $x{:}A$ and $x{:}\neg A$ are in S

A constraint system is *complete* if none of the above completion rules applies to it. A *clash* is a constraint of the form $x{:}\perp$. We say that S' is a *completion* of $\{x{:}C\}$, if S' is complete, and is obtained from $\{x{:}C\}$ by applying the above completion rules.

In [10] it is shown that an $\mathcal{AL}$-concept C is satisfiable iff the complete constraint system obtained from $\{x{:}C\}$ by means of the above rules does not contain any clash. Moreover, it is proved that computing the completion of $\{x{:}C\}$ is a polynomial task. By exploiting the features of the propagation rules, in [5] it is shown that checking subsumption between two $\mathcal{AL}$-concepts is also a polynomial task.

Concept languages can also be used as *assertional languages*, i.e. to make assertions on individual objects. Let O be an alphabet of symbols denoting individuals, and $\mathcal{L}$ be a concept language. An $\mathcal{L}$-assertion is a statement of one of the forms: $C(a)$, $R(a,b)$, where C is a concept of $\mathcal{L}$, R is a role of $\mathcal{L}$, and a, b are individuals in O. The meaning of the above assertions is straightforward: if $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation, $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $R(a,b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

A finite set Σ of $\mathcal{L}$-assertions is called an $\mathcal{L}$-*knowledge base*. An interpretation $\mathcal{I}$ is said to be a *model* of Σ if every assertion of Σ is satisfied by $\mathcal{I}$. Σ is said to be *satisfiable* if it admits a model. We say that Σ *logically implies* an assertion α (written $\Sigma \models \alpha$) if α is satisfied by every model of Σ.

The above propagation rules can be exploited for checking the satisfiability of an $\mathcal{AL}$-knowledge base Σ. The idea is that an $\mathcal{AL}$-knowledge base Σ can be translated into a constraint system, denoted by S_Σ, by replacing every assertion $C(a)$ with $a{:}C$, and every assertion $R(a,b)$ with aRb (see [6]). One can easily verify that, up to variable renaming, only one completion, denoted $COMP_{\mathcal{AL}}(\Sigma)$, can be derived from S_Σ. Notice that the constraints in $COMP_{\mathcal{AL}}(\Sigma)$ regards both individuals in O and variables in V. In the sequel, we use the term *object* as an abstraction for individual and variable. Moreover, if Z is either a knowledge base or a concept, we write dim_Z to denote the size of Z.

Theorem 2.1 *An $\mathcal{AL}$-knowledge base Σ is satisfiable iff $COMP_{\mathcal{AL}}(\Sigma)$ is clash-free. Moreover, $COMP_{\mathcal{AL}}(\Sigma)$ can be computed in polynomial time with respect to dim_Σ.*

3 Enriching the language of the subsumer

The goal of this section is to show that, when using a tractable language for the subsumee, it is possible to enrich the language of the subsumer without endangering the tractability of the subsumption problem. In particular, we study the subsumption problem (is C subsumed by D?) in the hypothesis that the candidate subsumee C is a concept of $\mathcal{AL}$, and the candidate subsumer D is a concept of a richer language, which we call $\mathcal{QL}$.

The language $\mathcal{QL}$ is defined by the following syntax (where P_i denotes a primitive role, and $n \geq 1$):

$$\begin{array}{rcl} C, D & \longrightarrow & A \mid \neg A \mid C \sqcap D \mid \{a_1, \ldots, a_n\} \mid \\ & & \forall R.C \mid \exists R \mid \exists R.C \\ R & \longrightarrow & P_1 \sqcap \cdots \sqcap P_n \end{array}$$

Notice that the results reported in [5] show that checking subsumption between two $\mathcal{QL}$-concepts is an NP-hard problem.

A concept C is subsumed by D iff $C \sqcap \neg D$ is unsatisfiable, thus we can reduce subsumption between a $\mathcal{QL}$-concept D and an $\mathcal{AL}$-concept C to unsatisfiability of $C \sqcap \neg D$. In order to solve such an unsatisfiability problem, we have devised suitable completion rules for $\mathcal{QL}$-constraint systems, i.e. constraint systems whose constraints have the forms: $x{:}C$, $x{:}\neg D$, and xRy, where C is an $\mathcal{AL}$-concept, D is a $\mathcal{QL}$-concept, and R is a $\mathcal{QL}$-role.

As a notation, we say that xRy *holds in* a constraint system S if: R is a primitive role and $xRy \in S$ or R is of the form $P_1 \sqcap \cdots \sqcap P_n$ and for each i, $xP_iy \in S$.

The set of completion rules for $\mathcal{QL}$-constraint systems is constituted by the rules for $\mathcal{AL}$ presented in Section 2, together with the following rules, that take care of the constructs of $\neg D$.

5. $S \rightarrow_{\neg\sqcap} \{x{:}\neg D_i\} \cup S$
   if $x{:}\neg(D_1 \sqcap D_2)$ is in S, $i \in \{1,2\}$,
   and neither $x{:}\neg D_1$ nor $x{:}\neg D_2$ is in S

6. $S \rightarrow_{\neg\forall} \{xP_1y, \ldots, xP_ny, y{:}\neg D\} \cup S^-$
   if $x{:}\neg\forall(P_1 \sqcap \cdots \sqcap P_n).D$ is in S, y is a
   new variable, and S^- is the constraint
   system obtained from S by replacing
   each variable z such that
   $xP_jz \in S$ ($j \in \{1, \ldots, n\}$) with y

7. $S \rightarrow_{\neg\exists} \{y{:}\neg D\} \cup S$
 if $x{:}\neg\exists R.D$ is in S, xRy holds in S, and $y{:}\neg D$ is not in S.

Observe that in the $\rightarrow_{\neg\forall}$-rule, each variable z previously created to satisfy one constraint of the form $x{:}\exists P_i$ ($i \in \{1,\ldots,n\}$) is replaced by the newly created variable y. This procedure, that is crucial for efficiency, is made possible by the fact that the existential quantification in $\mathcal{AL}$ is unqualified, and hence all the properties imposed on such z's are also imposed on y. It follows that it not necessary to keep track of the z's in the resulting system.

Notice that, due to the non-determinism of the $\rightarrow_{\neg\sqcap}$-rule, several complete constraint systems can be obtained from $\{x{:}C \sqcap \neg D\}$. The following theorem states the soundness and completeness of the above rules. Its proof derives from the above observation about the $\rightarrow_{\neg\forall}$-rule and from the results reported in [10].

Theorem 3.1 *Let C be an $\mathcal{AL}$-concept, and let D be a $\mathcal{QL}$-concept. Then $C \sqcap \neg D$ is unsatisfiable iff every completion of $\{x{:}C \sqcap \neg D\}$ contains a clash.*

It is easy to see that, starting from $\{x{:}C \sqcap \neg D\}$, in a finite number of applications of the rules, all the completions are computed, and checked for clash. It follows that the above propagation rules provide an effective procedure to check subsumption between D and C.

With regard to the computational complexity, the next theorem states that such a procedure requires polynomial time.

Theorem 3.2 *Let C be an $\mathcal{AL}$-concept, and let D be a $\mathcal{QL}$-concept. Then the set of all the completions of the constraint system $\{x{:}C \sqcap \neg D\}$ can be computed in polynomial time with respect to $dim_{C\sqcap\neg D}$.*

From all the above propositions it follows that checking subsumption between a $\mathcal{QL}$-concept and an $\mathcal{AL}$-concept can be done in polynomial time. This result will be exploited in next section for devising a polynomial query answering procedure.

4 Query answering

In this section we propose a query answering method that allows one to pose queries expressed in the language $\mathcal{QL}$ to an $\mathcal{AL}$-knowledge base.

As we said in the introduction, a query has the form of a concept D, and answering a query D posed to the knowledge base Σ means computing the set $\{a \in O \mid \Sigma \models D(a)\}$. In order to solve this problem, we consider the so-called *instance* problem: given an $\mathcal{AL}$-knowledge base Σ, a $\mathcal{QL}$-concept D, and an individual a, check if $\Sigma \models D(a)$. Since the number of individuals in Σ is finite, it is clear that our method can be directly used for query answering, in particular, by solving the instance problem for all the individuals in Σ.

Most of the existing approaches to the instance problem are based on the notion of most specialized concept (MSC). The MSC of an individual a is a representative of the complete set of concepts which a is an instance of. However, a method merely based on the MSC would not work in our case, because of the presence of the qualified existential quantification in $\mathcal{QL}$. For example, in order to answer the query $\exists R_1.\exists R_2.\{b,d\}(a)$, it is not sufficient to look at the MSC of a, but it is necessary to consider the assertions involving the roles R_1 and R_2 in the knowledge base. For this reason, our method relies on an ad hoc technique that, by navigating through the role assertions, takes into account the whole knowledge about the individuals.

In the following, we make use of a function ALL that, given an object a, a $\mathcal{QL}$-role $R = P_1 \sqcap \ldots \sqcap P_n$, and an $\mathcal{AL}$-knowledge base Σ, computes the $\mathcal{AL}$-concept $ALL(a,R,\Sigma) = C_1 \sqcap \cdots \sqcap C_m$, where $C_1,\ldots,C_m$ are all the concepts appearing in some constraint of $COMP_{\mathcal{AL}}(\Sigma)$ having the form $a{:}\forall Q.C_i$ with $Q \in \{P_1,\ldots,P_n\}$. If no such a concept exists, we assume $ALL(a,R,\Sigma) = \top$ (where $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$). In other words, $ALL(a,R,\Sigma)$ represents the concept to which every object related to a through R must belong, according to the assertions in Σ.

Our method heavily relies on the following theorem.

Theorem 4.1 *Let Σ be a satisfiable $\mathcal{AL}$-knowledge base, let $a, a_1,\ldots,a_n$ be individuals, let A be a primitive concept, and let D, D_1, D_2 be $\mathcal{QL}$-concepts. Then the following properties hold:*

1. $\Sigma \models \{a_1,\ldots,a_n\}(a)$ *iff* $a \in \{a_1,\ldots,a_n\}$;
2. $\Sigma \models A(a)$ *iff* $a{:}A \in COMP_{\mathcal{AL}}(\Sigma)$, *and* $\Sigma \models \neg A(a)$ *iff* $a{:}\neg A \in COMP_{\mathcal{AL}}(\Sigma)$;
3. $\Sigma \models D_1 \sqcap D_2(a)$ *iff* $\Sigma \models D_1(a)$ *and* $\Sigma \models D_2(a)$;
4. $\Sigma \models \forall R.D(a)$ *iff* D *subsumes* $ALL(a,R,\Sigma)$;
5. $\Sigma \models \exists R.D(a)$ *iff there is a* b *such that* aRb *holds in* $COMP_{\mathcal{AL}}(\Sigma)$ *and* $\Sigma \models D(b)$.

Proof. (Sketch) The proofs of 1, 2 and 3 are straightforward. With regard to 4, assume that D subsumes $ALL(a,R,\Sigma)$, and suppose that $\Sigma \not\models \forall R.D(a)$, i.e. $\Sigma \cup \{\exists R.\neg D(a)\}$ is satisfiable. This implies that there is a model $\mathcal{I}$ of Σ with an element $d \in \Delta^{\mathcal{I}}$ such that $d \in (ALL(a,R,\Sigma))^{\mathcal{I}}$, and $d \in (\neg D)^{\mathcal{I}}$, contradicting the hypothesis that $ALL(a,R,\Sigma)$ is subsumed by

```
**Algorithm** $ASK(\Sigma, a, D)$
**Input** $\mathcal{AL}$-knowledge base $\Sigma$, individual $a$,
        $\mathcal{QL}$-concept $D$, data structure $\mu$;
**Output** one value in $\{true, false\}$, updated $\mu$;
**begin**
  **if** $\mu(a, D) = nil$ **then**
  **case** $D$ **of**
    $A : \mu(a,D) \leftarrow a{:}A \in COMP_{\mathcal{AL}}(\Sigma)$;
    $\neg A : \mu(a,D) \leftarrow a{:}\neg A \in COMP_{\mathcal{AL}}(\Sigma)$;
    $D_1 \sqcap D_2 : \mu(a,D) \leftarrow ASK(\Sigma, a, D_1) \wedge$
                                  $ASK(\Sigma, a, D_2)$;
    $\forall R.D_1 : \mu(a,D) \leftarrow D_1$ subsumes $ALL(a, R, \Sigma)$;
    $\exists R.D_1 : \mu(a,D) \leftarrow \exists b$ s.t. $aRb$ holds in
                          $COMP_{\mathcal{AL}}(\Sigma) \wedge ASK(\Sigma, b, D_1)$;
    $\{a_1, \ldots, a_n\} : \mu(a,D) \leftarrow a \in \{a_1, \ldots, a_n\}$
  **endcase**
  **endif**;
  **return** $\mu(a, D)$
**end**
```

Figure 1: The Algorithm ASK

D. On the other hand, assume that $\Sigma \models \forall R.D(a)$, i.e. $\Sigma \cup \{\exists R.\neg D(a)\}$ is unsatisfiable, implying that $S_\Sigma \cup \{aRz, z{:}\neg D\}$ is unsatisfiable, where z is a new variable. Now, it is possible to verify that, since Σ is satisfiable, this may happen only because the constraint system $\{z{:}ALL(a,R,\Sigma), z{:}\neg D\}$ is unsatisfiable, which means that D subsumes $ALL(a,R,\Sigma)$. With regard to 5, it is easy to verify that if there is a b such that aRb holds in $COMP_{\mathcal{AL}}(\Sigma)$ and $\Sigma \models D(b)$, then $\Sigma \models \exists R.D(a)$. On the other hand, assume that $\Sigma \models \exists R.D(a)$, and suppose that for no b_i ($i = 1, \ldots, n$) such that aRb_i holds in $COMP_{\mathcal{AL}}(\Sigma)$, $\Sigma \models D(b_i)$. This implies that for each b_i, there is a model M_i of $\Sigma \cup \{\neg D(b_i)\}$. Now one can easily verify that $M_1 \cup \cdots \cup M_n$ is a model of $\Sigma \cup \{\forall R.\neg D(a)\}$, contradicting the hypothesis that $\Sigma \models \exists R.D(a)$. □

Based on the properties stated in the above theorem, we can directly develop an algorithm for query answering. The algorithm called ASK and shown in Fig. 1, makes use of a data structure μ which associates a value in $\{nil, true, false\}$ with every pair a', D', where a' is an object and D' is a $\mathcal{QL}$-concept. Informally speaking, $\mu(a', D')$ is used to record the answer to the query $\Sigma \models D'(a')$. The value nil represents that no answer has been yet computed for the query, whereas $true$ and $false$ have the obvious meaning of yes and no, respectively. We assume that, initially, $\mu(a', D') = nil$ for each pair a', D'.

The following theorem states the correctness and the tractability of the algorithm. Tractability is achieved by virtue of the data structure μ, which ensures that at most one call of the algorithm is issued for every pair a', D', where a' is an object and D' is a subconcept of D (i.e. a concept appearing in D). Notice that without a technique of this kind, the method might require an exponential number of checks (for example, when queries have the form $\exists R_1.\exists R_2.\ldots\exists R_n.A(a)$).

Theorem 4.2 *Let Σ be a satisfiable $\mathcal{AL}$-knowledge base, a be an individual, and D be a $\mathcal{QL}$-concept. Then $ASK(\Sigma, a, D)$ terminates, returning true if $\Sigma \models D(a)$, and false otherwise. Moreover, it runs in polynomial time with respect to dim_Σ and dim_D.*

5 Limits to the tractability of query answering

The aim of this section is to consider several possible extensions of both the query language and the assertional language and analyze their effect on the tractability of query answering.

We first consider the query language, showing that there are limits to its expressive power. The basic observation is that if D is equivalent to the universal concept $\top$, then for any knowledge base Σ, it holds that $\Sigma \models D(a)$. It follows that query answering is at least as hard as the so-called *top-checking* problem for the query language, i.e. checking whether a concept is equivalent to $\top$.

Notice that, due to the characteristics of $\mathcal{QL}$, for any $\mathcal{QL}$-concept D, $\neg D$ is satisfiable, and therefore in $\mathcal{QL}$ it is impossible to express a universal concept. However, there are languages in which the universal concept can be expressed, and in some of these languages top-checking is intractable. We are able to show that this is the case already for $\mathcal{FLU}^-$, that is obtained from $\mathcal{FL}^-$ simply by adding disjunction of concepts. The proof, reported in [7], is based on a reduction Φ from the satisfiability problem for a propositional conjunctive normal form (CNF) formula to the top-checking problem in $\mathcal{FLU}^-$. The reduction Φ is defined as follows:

$\Phi(p_i) = \exists R_{p_i}$, $\quad \Phi(\neg p_i) = \forall R_{p_i}.A$,
$\Phi(l_1 \vee \cdots \vee l_n) = \Phi(l_1) \sqcap \cdots \sqcap \Phi(l_n)$,
$\Phi(\alpha_1 \wedge \cdots \wedge \alpha_m) = \Phi(\alpha_1) \sqcup \cdots \sqcup \Phi(\alpha_m)$,

where p_i denotes a propositional letter, l_i a literal, and α_i a clause.

The above result allows us to derive the intractability of several other concept languages as query languages. For example, co-NP-hardness clearly extends to $\mathcal{ALU}$ ($\mathcal{AL}$ + disjunction), $\mathcal{ALC}$ ($\mathcal{FL}^-$ + full complement) [10], and $\mathcal{FL}$ [2].

We now turn our attention to analyzing possible extensions of the assertional language. Analogously to the case of the query language, we can single out an inherent limit to the expressive power of the assertional language, due to the fact that query answering is clearly at least as hard as the problem of concept satisfiability for the assertional language.

This observation allows us to rule out several extensions of $\mathcal{AL}$, such as $\mathcal{ALU}$, $\mathcal{ALE}$ ($\mathcal{AL}$ + qualified existential quantification), and $\mathcal{ALR}$ ($\mathcal{AL}$ + role conjunction) [5,10]. We are able to show that a similar result holds for the language $\mathcal{ALO}$, obtained from $\mathcal{AL}$ by adding collections of individuals. The proof, reported in [7], is based on a reduction Ψ from the NP-complete problem of checking the satisfiability of a CNF formula with only positive and negative clauses, to the problem of concept satisfiability in $\mathcal{ALO}$.

The reduction Ψ from $\Gamma = \alpha_1^+ \wedge \cdots \wedge \alpha_n^+ \wedge \alpha_1^- \wedge \cdots \wedge \alpha_m^-$ to the $\mathcal{ALO}$-concept $\Psi(\Gamma) = C_1^+ \sqcap \cdots \sqcap C_n^+ \sqcap C_1^- \sqcap \cdots \sqcap C_m^-$, is specified by the following equation:

$C_h^+ = \exists R_h^+ \sqcap \forall R_h^+.(obj(\alpha_h^+) \sqcap A)$,

$C_k^- = \exists R_k^- \sqcap \forall R_k^-.(obj(\alpha_k^-) \sqcap \neg A)$

where α_i^+ (α_i^-) denotes a positive (negative) clause, A is a primitive concept, R_h^+ and R_k^- (for $h = 1, \ldots, n$ and $k = 1, \ldots, m$) are primitive roles, and $obj(\alpha)$ denotes the concept $\{p_1, \ldots, p_k\}$, where $p_1, \ldots, p_k$ are all the propositional letters in the clause α. In other words, we associate with every propositional letter of Γ an individual with the same name, and with every clause α of Γ the collection of individuals $obj(\alpha)$.

For example, if $\Gamma = (p \vee q) \wedge (\neg p \vee \neg r)$, then the corresponding $\mathcal{ALO}$-concept $\Psi(\Gamma)$ is:

$\exists R_1^+ \sqcap \forall R_1^+.(\{p, q\} \sqcap A) \sqcap \exists R_1^- \sqcap \forall R_1^-.(\{p, r\} \sqcap \neg A)$.

Based on the above result, we are able to show that subsumption in CLASSIC [1] is intractable too[1]. Consider a primitive concept B and a primitive role R: it is easy to verify that the above reduction still holds when A and $\neg A$ are replaced by the two CLASSIC concepts (*AND B* (*ATLEAST* 2 *R*)) and (*AND B* (*ATMOST* 1 *R*)). It follows that concept satisfiability in CLASSIC is NP-hard, and therefore subsumption and query answering are co-NP-hard.

6 Conclusion

In the future, we aim at addressing several open problems related to the use of concept languages as query languages. First of all, we aim at investigating possible extensions of $\mathcal{AL}$ and $\mathcal{QL}$ (e.g. number restrictions), and we want to consider the case where the knowledge base includes a so-called terminology, i.e. an intentional part expressed in terms of concept definitions.

Second, we aim at improving the efficiency of our method for query answering, in particular by using a suitable extension of the notion of most specialized concept (see [4]) and by employing suitable techniques from the theory of query optimization in the relational data model. In fact, the goal of our work was to show that the problem is tractable, but several optimization of the algorithm are needed in order to cope with sizable knowledge bases.

Finally, we aim at considering more complex queries, such as queries constituted by a set of atomic assertions, or queries asking information regarding the intensional knowledge associated to the individuals.

[1]CLASSIC extends $\mathcal{FL}^-$ in various ways, including number restrictions and collections of individuals.

References

[1] A. Borgida, R. J. Brachman, D. L. McGuinness, L.A. Resnick. "CLASSIC: A Structural Data Model for Objects." *Proc. of ACM SIGMOD-89*, 1989.

[2] R. J. Brachman, H. J. Levesque. "The Tractability of Subsumption in Frame-based Description Languages." *Proc. of AAAI-84*, 1984.

[3] F. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, W. Nutt. "The Complexity of Existential Quantification in Terminological Reasoning", *Tech. Rep.* RAP.01.91, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1991.

[4] F. M. Donini, M. Lenzerini, D. Nardi, "An Efficient Method for Hybrid Deduction", *Proc. of ECAI-90*, 1990.

[5] F. Donini, M. Lenzerini, D. Nardi, W. Nutt. "The Complexity of Concept Languages." To appear in *Proc. of KR-91*, 1991.

[6] B. Hollunder, "Hybrid Inference in KL-ONE-based Knowledge Representation Systems." *German Workshop on Artificial Intelligence*, 1990.

[7] M. Lenzerini, A. Schaerf. "Concept Languages as Query Languages." *Technical Report*, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza". Forthcoming.

[8] H.J. Levesque. "The Interaction with Incomplete Knowledge Bases: a Formal Treatment." *Proc. of IJCAI-81*, 1981.

[9] B. Nebel. "Computational Complexity of Terminological Reasoning in BACK." *Artificial Intelligence*, **34**(3):371–383, 1988.

[10] M. Schmidt-Schauß, G. Smolka. "Attributive Concept Descriptions with Unions and Complements". To appear in *Artificial Intelligence*.