QUONTO: QUerying ONTOlogies

Andrea Acciarri[2], Diego Calvanese[1], Giuseppe De Giacomo[2], Domenico Lembo[2], Maurizio Lenzerini[2], Mattia Palmieri[2], Riccardo Rosati[2]

[1] Faculty of Computer Science
Free University of Bolzano/Bozen
Piazza Domenicani 3
I-39100 Bolzano, Italy
calvanese@inf.unibz.it

[2] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
I-00198 Roma, Italy
lastname@dis.uniroma1.it

Introduction

One of the most important lines of research in Description Logics (DLs) is concerned with the trade-off between expressive power and computational complexity of sound and complete reasoning. Research carried out in the past on this topic has shown that many DLs with efficient, i.e., worst-case polynomial time, reasoning algorithms lack the modeling power required for capturing conceptual models and basic ontology languages, while most DLs with sufficient modeling power suffer from inherently worst-case exponential time behavior of reasoning [1, 2].

Although the requirement of polynomially tractable reasoning might be less stringent when dealing with relatively small ontologies, we believe that the need of efficient reasoning algorithms is of paramount importance when the ontology system is to manage large amount of objects (e.g., from thousands to millions of instances). This is the case of several important applications where the use of ontologies is advocated nowadays. For example, in the Semantic Web, ontologies are often used to describe the relevant concepts of Web repositories, and such repositories may incorporate very large data sets, which constitute the instances of the concepts in the ontology. In such cases, two requirements emerge that are typically overlooked in DLs. First, the number of objects in the knowledge bases requires managing instances of concepts (i.e., ABoxes) in secondary storage. Second, significant queries to be posed to the knowledge base are more complex than the simple queries (i.e., concepts and roles) usually considered in DL research. Unfortunately, in these contexts, whenever the complexity of reasoning is exponential in the size of the instances (as for example in Fact[1], Racer[2] and in [3]), there is little hope for effective instance management and query answering algorithms.

In [4] a new DL, called *DL-Lite*, was proposed specifically tailored to capture basic ontology languages, while keeping low complexity of reasoning. A *DL-Lite* knowledge base (KB) is constituted by two components: an intensional level (called TBox in DL jargon), used to model the concepts and the relations (roles) of the ontologies, and an extensional level (ABox), used to represent instances of concepts and roles. Reasoning in *DL-Lite* means not only computing subsumption between concepts, and checking satisfiability of the whole knowledge base, but also answering complex queries. Notably, the complexity of answering queries posed to a knowledge base is polynomial in the size of the ABox.

In this work, we present QUONTO, a query answering system based on *DL-Lite*. Our system provides three basic functionalities: (1) specification of the intensional level of the ontology (TBox), (2) specification of the extensional level of the ontology (ABox), and (3) query answering. In the following, we describe the main characteristics of the system with respect to these three aspects.

TBox Specification

As usual in DLs, *DL-Lite* allows for representing the domain of interest in terms of concepts, denoting sets of objects, and roles, denoting binary relations between objects. *DL-Lite* concepts are defined as follows:

$$\begin{array}{rcl} B & ::= & A \mid \exists R \mid \exists R^- \\ C & ::= & B \mid \neg B \mid C_1 \sqcap C_2 \end{array}$$

where A denotes an atomic concept and R denotes an (atomic) role; B denotes a *basic concept* that can be either an atomic concept, a concept of the form $\exists R$, i.e., the standard DL construct of unqualified existential quantification on roles, or a concept of the form $\exists R^-$, which involves an *inverse role*. C (possibly with subscript) denotes a (general) concept. Note that we use negation of basic concepts only, and we do not allow for disjunction.

In QUONTO, the intensional level of the knowledge base is simply a *DL-Lite TBox*, i.e., a set of assertions of the form

$$\begin{array}{ll} B \sqsubseteq C & \textit{inclusion assertions} \\ (\mathsf{funct}\ R), (\mathsf{funct}\ R^-) & \textit{functionality assertions} \end{array}$$

An inclusion assertion expresses that a basic concept is subsumed by a general concept, while a functionality assertion expresses the (global) functionality of a role, or of the inverse of a role.

Despite the simplicity of its language, *DL-Lite* is able to capture the main notions (though not all, obviously) of conceptual modeling formalism used in databases and software engineering such as ER and UML class diagrams. In particular, *DL-Lite* assertions allow us to specify *ISA* and *disjointness* between concepts, *role-typing*, *participation* and

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[1] www.cs.man.ac.uk/~horrocks/FaCT

[2] www.sts.tu-harburg.de/~r.f.moeller/racer

non-participation constraints between a concept and a role, and *functionality restrictions* on roles.

ABox Specification

In QUONTO, the extensional level of the knowledge base is simply a *DL-Lite ABox*, i.e., a set of assertions of the form

$$A(c), R(c, b), \quad \textit{membership assertions}$$

where c and b are constants. These assertions state respectively that the object denoted by c is an instance of the atomic concept A, and that the pair of objects denoted by (c, b) is an instance of the role R.

One of the distinguishing feature of QUONTO is that the ABox is stored under the control of a DBMS, in order to effectively manage objects in the knowledge base by means of an SQL engine. To this aim, QUONTO constructs a relational database which faithfully represents an ABox $\mathcal{A}$: for each atomic concept A, a relational table tab_A of arity 1 is defined, such that $\langle c \rangle \in tab_A$ iff $A(c) \in \mathcal{A}$, and for each role R, a relational table tab_R of arity 2 is defined, such that $\langle c, b \rangle \in tab_R$ iff $R(c, b) \in \mathcal{A}$. We denote with $\mathsf{DB}(\mathcal{A})$ the relational database thus constructed.

Query answering

Perhaps, the main feature of our system is the ability to answer conjunctive queries posed to an ontology. While virtually all DL-based systems allow for answering atomic queries only (i.e., queries constituted by concepts or roles), QUONTO is able to answer conjunctive queries over a DL knowledge base.

A conjunctive query (CQ) q over a knowledge base $\mathcal{K}$ is an expression of the form

$$q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y})$$

where $\vec{x}$ are the so-called *distinguished variables*, $\vec{y}$ are existentially quantified variables called the *non-distinguished* variables, and $conj(\vec{x}, \vec{y})$ is a conjunction of atoms of the form $A(z)$, or $R(z_1, z_2)$, where A and R are respectively an atomic concept and a role in $\mathcal{K}$, and z, z_1, z_2 are onstants in $\mathcal{K}$ or variables in $\vec{x}$ or $\vec{y}$.

A conjunctive query $q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y})$ is interpreted in an interpretation $\mathcal{I}$ for $\mathcal{K}$ as the set $q^{\mathcal{I}}$ of tuples $\vec{c}$ such that when we substitute the variables $\vec{x}$ with the constants $\vec{c}$, the formula $\exists \vec{y}.conj(\vec{x}, \vec{y})$ evaluates to true in $\mathcal{I}$. Answering a conjunctive query q posed to a knowledge base $\mathcal{K}$ means computing the set of tuples $\vec{c}$ of constants of $\mathcal{K}$ such that in every model $\mathcal{I}$ of $\mathcal{K}$ we have $\vec{c} \in q^{\mathcal{I}}$.

Answering conjunctive queries over a knowledge base is a challenging problem, even in the case of *DL-Lite*, where the combination of allowable constructs does not pose particular difficulties in computing subsumption. Notice that, in spite of the simplicity of *DL-Lite* TBoxes, the ability of taking TBox knowledge into account during the process of answering conjunctive queries goes beyond the "variable-free" fragments of first-order logic represented by DLs.

In order to take advantage of the fact that the ABox is managed in secondary storage by a Data Base Management System (DBMS), our query answering algorithm is based on the idea of reformulating the original query into a set of queries that can be directly evaluated by an SQL engine over the ABox. Note that this allow us to take advantage of well established query optimization strategies.

Query reformulation is therefore at the heart of our query answering method. Given the limited expressive power of *DL-Lite* TBoxes, it might seem that in order to answer a query q over a KB $\mathcal{K}$ constituted by a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, we could simply build a finite first-order structure on the basis of $\mathcal{K}$, and then evaluate the query as an expression over this first-order structure. Actually, it is possible to show that this is not the case. In particular, it can be shown that, in general, given a KB $\mathcal{K}$, there exists no finite structure $\mathcal{S}$ such that, for every conjunctive query q, the set of answers to q over $\mathcal{K}$ is the result of evaluating q over $\mathcal{S}$. This property demonstrates that answering queries in *DL-Lite* goes beyond both propositional logic and relational databases. The basic idea of our method is to reformulate the query taking into account the TBox: in particular, given a query q over $\mathcal{K}$, we compile the assertions of the TBox into the query itself, thus obtaining a new query q'. Such a new query q' is then evaluated over the ABox of $\mathcal{K}$, as if the ABox were a simple relational database. Since the size of q' does not depend on the ABox, the data complexity of the whole query answering algorithm is polynomial.

Finally, we observe that query answering can be used in QUONTO for other forms of reasoning on the knowledge base $\mathcal{K}$. For example, to check whether $\mathcal{K}$ is unsatisfiable, we can simply add the assertion $A(c)$ to the Abox (where c is new constant), the inclusion $A \sqsubseteq \neg D$ to the TBox, and check whether c is in the answer to the query $q(x) \leftarrow D(x)$. Similarly, to check whether $\mathcal{K} \models A \sqsubseteq C$, we can simply add the assertion $A(c)$ to the Abox (where c is new constant), and check whether c is in the answer to the query $q(x) \leftarrow C'(x)$, where C' is the conjunctive query corresponding to the concept C.

Conclusions

Our experiments on QUONTO are extremely encouraging. The system is able to efficiently answer complex conjunctive queries (actually, unions of conjunctive queries) over ABoxes constituted by hundreds of thousands of instances of the concepts in the TBox. To the best of our knowledge, this is the first system exhibiting the ability to effectively answer complex queries over ontologies.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [1], chapter 10, pages 349–372.

[3] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of AAAI 2000*, pages 386–391, 2000.

[4] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical reasoning for rich DLs. In *Proc. of DL 2004*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-104/`, 2004.