Ontology-based database access

Diego Calvanese[1], Giuseppe De Giacomo[2], Domenico Lembo[2],
Maurizio Lenzerini[2], Antonella Poggi[2], Riccardo Rosati[2]

[1]Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3
I-39100 Bolzano, Italy
[2]Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
I-00198 Roma, Italy
[1]calvanese@inf.unibz.it,[2]{lastname}@dis.uniroma1.it

Abstract. One of the most interesting usages of shared conceptualizations is *ontology-based data access*. That is, to the usual data layer of an information system we superimpose a conceptual layer to be exported to the client. Such a layer allows the client to have a conceptual view of the information in the system, which abstracts away from how such information is maintained in the data layer of the system itself. While ontologies are the best candidates for realizing the conceptual layer, relational DBMSs are natural candidates for the management of the data layer. The need of efficiently processing large amounts of data requires ontologies to be expressed in a suitable fragment of OWL: the fragment should allow, on the one hand, for modeling the kind of intensional knowledge needed in real-world applications, and, on the other hand, for delegating to a relational DBMS the part of reasoning (in particular query answering) that deals with the data. In this paper, we propose one such a fragment, in fact the largest fragment currently known to satisfy the above requirements. Furthermore, we provide means to access databases that are independent from the ontology, by proposing a novel mapping language that solves the so-called "impedance mismatch" between values in the databases and objects represented in the ontology.

1 Introduction

In several areas, such as Enterprise Application Integration, Data Integration [9], and the Semantic Web [6], clients need to access the services exported by the system, and hence require a representation of the intensional level of the application domain in terms of which they can specify the access to the exported services. One of the most interesting usages of such a shared conceptualization is *ontology-based data access*, where a conceptual layer is exported to the client, abstracting away from how actual data is maintained. While ontologies are the best candidates for realizing the conceptual layer, relational DBMSs are natural candidates for the management of the data layer, since relational database technology is nowadays the best technology for efficient management of very large quantities of data.

Recently, basic research has shown that none of the variants of OWL is suitable to act as the formalism for representing ontologies in this context [4, 11, 8], if not re-

stricted (they all are coNP-hard w.r.t. data complexity). Possible restrictions that guarantee polynomial reasoning (at least, if we concentrate on instance checking only) have been looked at, such as Horn-$\mathcal{SHIQ}$ [8], $\mathcal{EL}^{++}$ [2], DLP [5]. Among such fragments, of particular interest are those belonging to the DL-Lite family [3, 4]. These logics allow for answering complex queries (namely, conjunctive queries, i.e., SQL select-project-join queries, and unions of conjunctive queries) in LOGSPACE w.r.t. data complexity (i.e., the complexity measured only w.r.t. the size of the data). More importantly, they allow for delegating query processing, after a preprocessing phase which is independent of the data, to the relational DBMS managing the data layer.

In this paper, we propose to use a new DL, called *DL-Lite*$_{\mathcal{A}}^{+}$, which keeps the above mentioned features of the other languages in the *DL-Lite* family, while allowing to distinguish between objects and values, by introducing, besides concepts and roles, also concept-attributes and role-attributes, that describe properties of concepts (resp., roles) represented by values rather than objects. Then, we look at the problem of accessing databases that are independent from the ontology. Observe, however, that such databases, being relational, store only values, not objects. Hence, to deal with the so-called "impedance mismatch", we propose to relate database values to the ontology by using a novel mapping language [9], such that objects are constructed from such values.

2 The description logic *DL-Lite*$_{\mathcal{A}}^{+}$

In this section we present a new logic of the *DL-Lite* family, called *DL-Lite*$_{\mathcal{A}}^{+}$. As usual in DLs, all logics of the *DL-Lite* family allow one to represent the universe of discourse in terms of concepts, denoting sets of objects, and roles, denoting binary relations between objects. In addition, the DLs discussed in this paper allow one to use (*i*) value-domains, a.k.a. concrete domains [10], denoting sets of (data) values, (*ii*) concept attributes, denoting binary relations between objects and values, and (*iii*) role attributes, denoting binary relations between pairs of objects and values. Obviously, a role attribute can be also seen as a ternary relation relating two objects and a value.

We first introduce the DL *DL-Lite*$_{\mathcal{FR}}$, that combines the main features of two DLs presented in [4], called *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, respectively, and forms the basics of *DL-Lite*$_{\mathcal{A}}^{+}$. The value-domains that we consider in *DL-Lite*$_{\mathcal{FR}}$ are those corresponding to the data types adopted by the Resource Description Framework (RDF)[1]. Intuitively, these types represent sets of values that are pairwise disjoint. In the following, we denote such value-domains by $T_1, \ldots, T_n$. Furthermore, we denote with Γ the alphabet for constants, which we assume partitioned into two sets, namely, Γ_V (the set of constant symbols for values), and Γ_O (the set of constant symbols for objects). In turn, Γ_V is partitioned into n sets $\Gamma_{V_1}, \ldots, \Gamma_{V_n}$, where each Γ_{V_i} is the set of constants for the values in the value-domain T_i.

In providing the specification of our logics, we use the following notation: A denotes an *atomic concept*, B a *basic concept*, C a *general concept*, and $\top_C$ the *universal concept*; E denotes a basic value-domain, i.e., the range of an attribute, F denotes a *general value-domain*, and $\top_D$ the *universal value-domain*; P denotes an *atomic role*, Q a *basic role*, and R a *general role*; U_C denotes an *atomic attribute*, and V_C a *general attribute*; U_R denotes an *atomic role attribute*, and V_R a *general role attribute*.

[1] `http://www.w3.org/RDF/`

Given a concept attribute U_C (resp. a role attribute U_R), we call *domain* of U_C (resp. U_R), denoted by $\delta(U_C)$ (resp. $\delta(U_R)$), the set of objects (resp. of pairs of objects) that U_C (resp. U_R) relates to values, and we call *range* of U_C (resp. U_R), denoted by $\rho(U_C)$ (resp. $\rho(U_R)$), the set of values that U_C (resp. U_R) relates to objects (resp. pairs of objects). Notice that the domain $\delta(U_C)$ of a concept attribute U_C is a concept, whereas the domain $\delta(U_R)$ of a role attribute U_R is a role.

In particular, *DL-Lite*$_{\mathcal{FR}}$ expressions are defined as follows.

- Basic and general concept expressions:

$$B \ ::= \ A \mid \exists Q \mid \delta(U_C), \qquad C \ ::= \ \top_C \mid B \mid \neg B$$

- Basic and general value-domain expressions:

$$E \ ::= \ \rho(U_C) \mid \rho(U_R), \qquad F \ ::= \ \top_D \mid T_1 \mid \cdots \mid T_n$$

- General concept and role attribute expressions:

$$V_C \ ::= \ U_C \mid \neg U_C, \qquad V_R \ ::= \ U_R \mid \neg U_R$$

- Basic and general role expressions:

$$Q \ ::= \ P \mid P^- \mid \delta(U_R) \mid \delta(U_R)^-, \qquad R \ ::= \ Q \mid \neg Q$$

A *DL-Lite*$_{\mathcal{FR}}$ *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is constituted by two components: a TBox $\mathcal{T}$, used to represent intensional knowledge, and an ABox $\mathcal{A}$, used to represent extensional knowledge. *DL-Lite*$_{\mathcal{FR}}$ *TBox* assertions are of the form:

$B \sqsubseteq C$	*concept inclusion*	(funct P)	*role functionality*
$Q \sqsubseteq R$	*role inclusion*	(funct P^-)	*inverse role functionality*
$E \sqsubseteq F$	*value-domain inclusion*	(funct U_C)	*concept attribute functionality*
$U_C \sqsubseteq V_C$	*concept attribute inclusion*	(funct U_R)	*role attribute functionality*
$U_R \sqsubseteq V_R$	*role attribute inclusion*		

A concept inclusion assertion expresses that a (basic) concept B is subsumed by a (general) concept C. Analogously for the other types of inclusion assertions. A role functionality assertion expresses the (global) functionality of an atomic role. Analogously for the other types of functionality assertions.

A *DL-Lite*$_{\mathcal{FR}}$ ABox is a finite set of assertions of the form:

$$A(a), \quad P(a, b), \quad U_C(a, b), \quad U_R(a, b, c)$$

where a, b and c are constants in the alphabet Γ.

Following the classical approach in DLs, the semantics of *DL-Lite*$_{\mathcal{FR}}$ is given in terms of first-order logic interpretations. All such intepretations agree on the semantics assigned to each value-domain T_i and to each constant in Γ_V. In particular, each T_i is interpreted as the set $val(T_i)$ of values of the corresponding RDF data type, and each $c_i \in \Gamma_V$ is interpreted as one specific value, denoted $val(c_i)$, in $val(T_i)$. Note that, for $i \neq j$, it holds that $val(T_i) \cap val(T_j) = \emptyset$.

Based on the above observations, we can now define the notion of interpretation in *DL-Lite*$_{\mathcal{FR}}$. An *interpretation* is a pair $I = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

– $\Delta^{\mathcal{I}}$ is the interpretation domain, that is the disjoint union of two sets: $\Delta^{\mathcal{I}}_O$, called the *domain of objects*, and $\Delta^{\mathcal{I}}_V$, called the *domain of values*. In turn, $\Delta^{\mathcal{I}}_V$ is the union of $val(T_1), \ldots, val(T_n)$.
– $\cdot^{\mathcal{I}}$ is the *interpretation function*, i.e., a function that assigns an element of $\Delta^{\mathcal{I}}$ to each constant in Γ,
  - for each $a \in \Gamma_V$, $a^{\mathcal{I}} = val(a)$,
  - for each $a \in \Gamma_O$, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}_O$,
  - for each $a, b \in \Gamma$, $a \neq b$ implies $a^{\mathcal{I}} \neq b^{\mathcal{I}}$,
  - for each T_i, $T_i^{\mathcal{I}} = val(T_i)$,
  - the following conditions are satisfied:

$$\top^{\mathcal{I}}_C = \Delta^{\mathcal{I}}_O \qquad (P^-)^{\mathcal{I}} = \{\ (o, o') \mid (o', o) \in P^{\mathcal{I}}\ \}$$
$$\top^{\mathcal{I}}_D = \Delta^{\mathcal{I}}_V \qquad (\rho(U_C))^{\mathcal{I}} = \{\ v \mid \exists o.\, (o, v) \in U^{\mathcal{I}}_C\ \}$$
$$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_O \qquad (\rho(U_R))^{\mathcal{I}} = \{\ v \mid \exists o, o'.\, (o, o', v) \in U^{\mathcal{I}}_R\ \}$$
$$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_O \qquad (\delta(U_C))^{\mathcal{I}} = \{\ o \mid \exists v.\, (o, v) \in U^{\mathcal{I}}_C\ \}$$
$$U^{\mathcal{I}}_C \subseteq \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_V \qquad (\delta(U_R))^{\mathcal{I}} = \{\ (o, o') \mid \exists v.\, (o, o', v) \in U^{\mathcal{I}}_R\ \}$$
$$U^{\mathcal{I}}_R \subseteq \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_V \qquad (\exists Q)^{\mathcal{I}} = \{\ o \mid \exists o'.\, (o, o') \in Q^{\mathcal{I}}\ \}$$
$$(\neg U_C)^{\mathcal{I}} = (\Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_V) \setminus U^{\mathcal{I}}_C \qquad (\neg Q)^{\mathcal{I}} = (\Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_O) \setminus Q^{\mathcal{I}}$$
$$(\neg U_R)^{\mathcal{I}} = (\Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_O \times \Delta^{\mathcal{I}}_V) \setminus U^{\mathcal{I}}_R \qquad (\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}}_O \setminus B^{\mathcal{I}}$$

Note that the above definition implies that different constants are interpreted differently in the domain, i.e., *DL-Lite*$_{\mathcal{FR}}$ adopts the so-called unique name assumption.

We define when an interpretation $\mathcal{I}$ satisfies an assertion (i.e., is a model of it) as follows (below, each o, possibly with subscript, is an element of $\Delta^{\mathcal{I}}$, and a, b and c are constants in Γ). Specifically, an interpretation $\mathcal{I}$ *satisfies* (*i*) an inclusion assertion $\alpha \sqsubseteq \beta$, if $\alpha^{\mathcal{I}} \subseteq \beta^{\mathcal{I}}$; (*ii*) a functional assertion (funct γ), where γ is either P, P^-, or U_C, if, for each o_1, o_2, o_3, $(o_1, o_2) \in \gamma^{\mathcal{I}}$ and $(o_1, o_3) \in \gamma^{\mathcal{I}}$ implies $o_2 = o_3$; (*iii*) a functional assertion (funct U_R), if for each o_1, o_2, o_3, o_4, $(o_1, o_2, o_3) \in U^{\mathcal{I}}_R$ and $(o_1, o_2, o_4) \in U^{\mathcal{I}}_R$ implies $o_3 = o_4$; (*iv*) a membership assertion $\alpha(a)$, where α is either A or D, if $a^{\mathcal{I}} \in \alpha^{\mathcal{I}}$; (*v*) a membership assertion $\beta(a, b)$, where β is either P or U_C, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in \beta^{\mathcal{I}}$; (*vi*) a membership assertion $U_R(a, b, c)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}, c^{\mathcal{I}}) \in U^{\mathcal{I}}_R$. A *model of a KB* $\mathcal{K}$ is an interpretation $\mathcal{I}$ that is a model of all assertions in $\mathcal{K}$. A KB is *satisfiable* if it has at least one model.

A conjunctive query (CQ) q over a knowledge base $\mathcal{K}$ is an expression of the form $q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}.conj(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ are the so-called *distinguished variables*, $\boldsymbol{y}$ are existentially quantified variables called the *non-distinguished* variables, and $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form $A(x)$, $D(x)$, $P(x, y)$, $U_C(x, y)$, $U_R(x, y, z)$, or $x = y$, where x, y, z are either variables in $\boldsymbol{x}$ or in $\boldsymbol{y}$ or constants in Γ. A *union of conjunctive queries* (UCQ) is a query of the form $q(\boldsymbol{x}) \leftarrow \bigvee_i \exists \boldsymbol{y_i}.conj(\boldsymbol{x}, \boldsymbol{y_i})$ A query $q(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x})$ is interpreted in $\mathcal{I}$ as the set $q^{\mathcal{I}}$ of tuples $\boldsymbol{o} \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}$ such that, when we assign $\boldsymbol{o}$ to the variables $\boldsymbol{x}$, the formula $\varphi(\boldsymbol{x})$ evaluates to true in $\mathcal{I}$.

The reasoning service we are interested in is *query answering*: given a knowledge base $\mathcal{K}$ and a UCQ $q(\boldsymbol{x})$ over $\mathcal{K}$, return the *certain answers* to $q(\boldsymbol{x})$ over $\mathcal{K}$, i.e., all tuples $\boldsymbol{t}$ of elements of Γ such that for every model $\mathcal{I}$ of $\mathcal{K}$.

From the results in [4] it follows that, in general, query answering over *DL-Lite*$_{\mathcal{FR}}$ KBs is PTIME-hard in data complexity (i.e., the complexity measured w.r.t. the size of the ABox only). As a consequence, to solve query answering over *DL-Lite*$_{\mathcal{FR}}$ KBs, we

need at least the power of general recursive Datalog. Since we are interested in DLs where query answering can be done in LOGSPACE, we introduce the DL *DL-Lite*$_{\mathcal{A}}^{+}$, which is obtained from *DL-Lite*$_{\mathcal{FR}}$ by imposing a limitation on the use of the functionality assertions in the TBox, as described next.

Definition 1. *A DL-Lite*$_{\mathcal{A}}^{+}$ *knowledge base is pair* $\langle \mathcal{T}, \mathcal{A} \rangle$*, where* $\mathcal{A}$ *is a DL-Lite*$_{\mathcal{FR}}$ *ABox, and* $\mathcal{T}$ *is a DL-Lite*$_{\mathcal{FR}}$ *TBox satisfying the following conditions:*

1. *for every role inclusion assertion* $Q \sqsubseteq R$ *in* $\mathcal{T}$*, where* R *is an atomic role or the inverse of an atomic role, the assertions* $(\mathsf{funct}\ R)$ *and* $(\mathsf{funct}\ R^{-})$ *are not in* $\mathcal{T}$*;*
2. *for every concept attribute inclusion assertion* $U_C \sqsubseteq V_C$ *in* $\mathcal{T}$*, where* V_C *is an atomic concept attribute, the assertion* $(\mathsf{funct}\ V_C)$ *is not in* $\mathcal{T}$*;*
3. *for every role attribute inclusion assertion* $U_R \sqsubseteq V_R$ *in* $\mathcal{T}$*, where* V_R *is an atomic role attribute, the assertion* $(\mathsf{funct}\ V_R)$ *is not in* $\mathcal{T}$*.*

Roughly speaking, a *DL-Lite*$_{\mathcal{A}}^{+}$ TBox imposes the condition that *every functional role cannot be specialized* by using it in the right-hand side of role inclusion assertions; the same condition is also imposed on every functional (role or concept) attribute.

In fact, it turns out that the above restriction is necessary in order to perform query answering over a *DL-Lite*$_{\mathcal{A}}^{+}$ ontology by following a technique similar to the one developed for the other logics in the *DL-Lite* family [3]. In particular, it can be shown [12] that query answering can be reduced to the evaluation of a first-order query over a relational database representing the ontology ABox. Such query is obtained by reformulating the original query based on the TBox assertions. Notably, this reformulation does not depend on the data, and hence query answering is LOGSPACE in data complexity.

3 Linking data to *DL-Lite*$_{\mathcal{A}}^{+}$ ontologies

Most work on DLs do not deal with the problem of how to store ABox assertions, nor do they address the issue of how to acquire ABox assertions from existing data sources. It is our opinion that this topic is of special importance in several contexts where the use of ontologies is advocated, especially in the case where the ontology is used to provide a unified conceptual model of an organization (e.g., in Enterprise Application Integration). In these contexts, the problem can be described as follows: the ontology is a virtual representation of a universe of discourse, and the instances of concepts and roles in the ontology are simply an abstract representation of some real data stored in existing data sources. Therefore, the problem arises of establishing sound mechanisms for linking existing data to the instances of the concepts and the roles in the ontology.

In this section we sketch our solution, by presenting a mapping mechanism that enables a designer to link data sources to an ontology expressed in *DL-Lite*$_{\mathcal{A}}^{+}$. Before delving into the details of the method, a preliminary discussion on the notorious impedance mismatch problem between data and objects is in order. When mapping data sources to ontologies, one should take into account that sources store data, whereas instances of concepts are objects, where each object should be denoted by an ad hoc identifier (e.g., a constant in logic), not to be confused with any data item. In *DL-Lite*$_{\mathcal{A}}^{+}$, we address this problem by keeping data value constants separate from object identifiers, and by accepting that object identifiers be created using data values, in particular as (logic)

terms over data items. Note that this idea traces back to the work done in deductive object-oriented databases [7].

To realize this idea, we modify the set Γ_O as follows. We assume that data appearing at the sources are denoted by constants in Γ_V, and we introduce a new alphabet Λ of function symbols in *DL-Lite*$_{\mathcal{A}}^{+}$, where each function symbol has an associated arity, specifying the number of argument it accepts. On the basis of Γ_V and Λ, we inductively define the set $\tau(\Lambda, \Gamma_V)$ of all *terms* of the form $f(d_1, \ldots, d_n)$ such that (*i*) $f \in \Lambda$, (*ii*) the arity of f is $n > 0$, and (*iii*) $d_1, \ldots, d_n \in \Gamma_V$. We finally sanction that the set Γ_O of symbols used in *DL-Lite*$_{\mathcal{A}}^{+}$ for denoting objects actually coincides with $\tau(\Lambda, \Gamma_V)$. In other words, we use the terms built out of Γ_V using the function symbols in Λ for denoting the instances of concepts in *DL-Lite*$_{\mathcal{A}}^{+}$ ontologies.

All the notions defined for our logics remain unchanged. In particular, an interpretation $I = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ still assigns a different element of $\Delta^{\mathcal{I}}$ to every element of Γ, which means that different terms in $\tau(\Lambda, \Gamma_V)$ are interpreted as different objects in $\Delta_O^{\mathcal{I}}$, i.e., we enforce the unique name assumption on terms.

Let us now turn our attention to the problem of linking data in the sources to objects in the ontology. To this aim, as we said before, we assume that all value constants stored in DB belong to Γ_V, and that the data sources are wrapped into a relational database DB (constituted by the relational schema, and the extensions of the relations), so that we can query such data by using SQL. Then, we adapt principles and techniques from the literature on data integration [9]. In particular, we use the notion of *mapping*, which we now introduce by means of an example.

Example 1. Consider a *DL-Lite*$_{\mathcal{A}}^{+}$ TBox in which *person* is a concept name, $\mathbf{age}$ and $\mathbf{cityName}$ are concept attributes names, *CITY-OF-BIRTH* is a role name, and a relational database contains the ternary relation symbols *S1* and *S2* and the unary relation symbol *S3*. We want to model the situation where every tuple $(n, s, a) \in$ *S1* corresponds to a person whose name is n, whose surname is s, and whose age is a, and we want to denote such a person with $p(n, s)$. Note that this implies that we know that there are no two persons in our application that have the same pair (n, s) stored in *S1*. Similarly, we want to model the fact that every tuple $(n, s, cb) \in$ *S2* corresponds to a person whose name is n, whose surname is s, and whose city of birth is cb. Finally, we know that source *S3* directly stores object constants denoting instances of person. The following is the set of mapping assertions modeling the above situation.

$$\begin{aligned}
&\textit{S1}(n, s, a) \rightsquigarrow \textit{person}(p(n, s)), \mathbf{age}(p(n, s), a)\\
&\textit{S2}(n, s, cb) \rightsquigarrow \textit{CITY-OF-BIRTH}(p(n, s), ct(cb)), \mathbf{cityName}(ct(cb), cb)\\
&\textit{S3}(q) \rightsquigarrow \textit{person}(q).
\end{aligned}$$

Above, n, s, a, cb and q are variable symbols, p and ct are function symbols, whereas $p(n, s)$ and $ct(n)$ are so-called variable terms (see below). ∎

The example shows that, in specifying mapping assertions, we need variable terms, i.e., terms containing variables. Indeed, we extend terms to *variable terms* of the form $f(\boldsymbol{z})$, where f is a function symbol in Λ of arity m, and $\boldsymbol{z}$ denotes an m-tuple of variables.

We can now provide the definition of mapping assertions. Through a mapping we associate a conjunctive query over atomic concepts, domains, roles, attributes, and role

attributes (generically referred to as *predicates* in the following) with a first-order (more precisely, SQL) query of the appropriate arity over the database. The intuition is that, by evaluating such a query, we retrieve the facts that constitute the ABox assertions for the predicates appearing in the conjunctive query. Formally, a *mapping assertion* is an assertion of the form: $\varphi \rightsquigarrow \psi$, where φ is an arbitrary SQL query of arity $n > 0$ over DB, and ψ is a UCQ over $\mathcal{T}$ of arity $n' > 0$ without non-distinguished variables, that possibly involves variable terms.

We now describe the semantics of mapping assertions. To this end, we introduce the notion of ground instance of a formula. Let $\gamma(\boldsymbol{x})$ be a formula over a *DL-Lite*$_{\mathcal{A}}^{+}$ TBox with n distinguished variables $\boldsymbol{x}$, and let $\boldsymbol{v}$ a tuple of value constants of arity n. Then the ground instance $\gamma[\boldsymbol{x}/\boldsymbol{v}]$ of $\gamma(\boldsymbol{x})$ is the formula obtained by substituting every occurrence of x_i with v_i (for $i \in \{1, .., n\}$) in $\psi(\boldsymbol{x})$. Let M be a mapping assertion of the form $\varphi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{t}, \boldsymbol{y})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are variables, $\boldsymbol{y} \subseteq \boldsymbol{x}$ and $\boldsymbol{t}$ are variable terms of the form $f(\boldsymbol{z})$, $f \in \Lambda$ and $\boldsymbol{z} \subseteq \boldsymbol{x}$. We say that I satisfies M with respect to a database DB, if for every tuple of values $\boldsymbol{v}$ such that $\boldsymbol{v} \in ans(\varphi, DB)$, and for each ground atom X in $\psi[\boldsymbol{x}/\boldsymbol{v}]$, we have that: (*i*) if X has the form $\alpha(s)$, where α is either A or D, then $s^{\mathcal{I}} \in \alpha^{\mathcal{I}}$; (*ii*) if X has the form $\beta(s_1, s_2)$, where β is either P or U_C, then $(s_1^{\mathcal{I}}, s_2^{\mathcal{I}}) \in \beta^{\mathcal{I}}$; (*iii*) if X has the form $U_R(s_1, s_2, s_3)$, then $(s_1^{\mathcal{I}}, s_2^{\mathcal{I}}, s_3^{\mathcal{I}}) \in U_R^{\mathcal{I}}$.

Finally, we can summarize the semantics of a *DL-Lite*$_{\mathcal{A}}^{+}$ ontology with mapping assertions, denoted with $\langle \mathcal{T}, \mathcal{M}, DB \rangle$, where DB is a database as defined above, $\mathcal{T}$ is a *DL-Lite*$_{\mathcal{A}}^{+}$ TBox, and $\mathcal{M}$ a set of mapping assertions between DB and $\mathcal{T}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a *model* of if $\mathcal{I}$ is a model of $\mathcal{T}$ and satisfies all mapping assertions in $\mathcal{M}$ wrt DB. The notion of certain answer to queries posed to $\langle \mathcal{T}, \mathcal{M}, DB \rangle$ remains the same as the one described in the previous section.

We now briefly sketch the technique for query answering over a *DL-Lite*$_{\mathcal{A}}^{+}$ ontology with mappings. First, we split each mapping assertion $\varphi \rightsquigarrow \psi$ into several assertions of the form $\varphi \rightsquigarrow p$, one for each atom p in ψ. Then, we unify in all possible ways the atoms in the query q to be evaluated with the right-hand side atoms of the (split) mappings, thus obtaining a (bigger) union of conjunctive queries containing variable object terms. Then, we unfold each atom with the corresponding left-hand side mapping query. Observe that, after unfolding, we obtain an SQL query that can be evaluated over DB, and possibly returns terms built from values extracted from DB.

Example 2. Refer to the previous example, and consider now the following query over the TBox, asking for the age of those people that are born in Rome:

$$q(z) \leftarrow \exists x, y. \, \mathit{person}(x), \mathit{CITY\text{-}OF\text{-}BIRTH}(x, y), \mathbf{cityName}(y, \mathsf{Roma}), \mathbf{age}(x, z)$$

Let us, for simplicity, assume that no reasoning on the TBox has to be done in order to answer the query q, and hence let us directly evaluate such a query by exploiting the mapping, without materializing the ABox of the KB.

We first split the mapping (left as an exercise), and then unify the atoms in the query with the right-hand side atoms in the split mapping, thus obtaining

$$\begin{aligned} q(z) \leftarrow \; & \mathit{person}(p(n, s)), \mathit{CITY\text{-}OF\text{-}BIRTH}(p(n, s), ct(\mathsf{Roma})), \\ & \mathbf{cityName}(ct(\mathsf{Roma}), \mathsf{Roma}), \mathbf{age}(p(n, s), z). \end{aligned}$$

Then, we unfold each atom with the corresponding left-hand side of the mapping query, and obtain the query: $q(z) \leftarrow S2(n, s, \mathsf{Roma}), S1(n, s, z)$, which can be simply evaluated over the database to get the certain answers to q. ∎

4 Conclusions

We argue that, for ontology-based data access, ontologies need to be expressed in a fragment of OWL that is LOGSPACE in data complexity, and that allows for delegating to the relational DBMS managing the data layer the part of reasoning that deals with the data. We have proposed one such fragment, *DL-Lite*$_{\mathcal{A}}^{+}$, which is in fact the biggest fragment currently known to satisfy the above requirements. In this paper we have looked at binary roles only, but all the results presented here can be extended to relations of arbitrary arity. All features introduced in this paper, have been implemented in the QuOnto system[2] [1] (originally based on a DL, called *DL-Lite*$_{\mathcal{F}}$, that is a subset of *DL-Lite*$_{\mathcal{A}}^{+}$).

Acknowledgments. This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU under contract number FP6-7603, by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant, and by MIUR FIRB 2005 project "Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet" (TOCAI.IT).

References

1. A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: QUerying ONTOlogies. In *Proc. of AAAI 2005*, pages 1670–1671, 2005.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI 2005*, pages 364–369, 2005.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, pages 602–607, 2005.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
5. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of WWW 2003*, pages 48–57, 2003.
6. J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.
7. R. Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases*, pages 193–256. Academic Press, 1988.
8. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI 2005*, pages 466–471, 2005.
9. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.
10. C. Lutz. Description logics with concrete domains: A survey. In P. Balbiani, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logics*, volume 4. King's College Publications, 2003.
11. M. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of AAAI 2006*, 2006.
12. A. Poggi. *Structured and Semi-Structured Data Integration*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 2006.

[2] `http://www.dis.uniroma1.it/~quonto/`