Linking Data to Ontologies: The Description Logic *DL-Lite*$_A$

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
calvanese@inf.unibz.it

[2] Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
I-00198 Roma, Italy
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.

1 Introduction

In several areas (e.g., Enterprise Application Integration, Data Integration [9], and the Semantic Web [6]) the intensional level of the application domain can be profitably represented by an ontology, so that clients can rely on a shared conceptualization when accessing the services provided by the system. One of the most interesting usages of such 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, since relational database technology is nowadays the best technology for efficient management of very large quantities of data.

Recently, basic research has been done in understanding which fragments of OWL, OWL-DL, or OWL-Lite would be suited to act as the formalism for representing ontologies in this context [4, 11, 8]. The outcome of this work is that none of the variants of OWL is suitable, if not restricted (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.

According to [4] there are two maximal languages in the DL-Lite family that possess the above property. The first one is *DL-Lite*$_F$, which allows for specifying the main modeling features of conceptual models, including cyclic assertions, ISA on concepts, inverses on roles, role typing, mandatory participation to roles, and functional restrictions on roles. The second one is *DL-Lite*$_R$, which is able to fully capture (the DL fragment of) RDFS, and has in addition the ability of specifying mandatory participation on roles and disjointness between concepts and roles. The language obtained by unrestrictedly merging the features of *DL-Lite*$_F$ and *DL-Lite*$_R$, while quite interesting in general, is not in LOGSPACE w.r.t. data complexity anymore [4], and hence loses the most interesting computational feature for ontology-based data access.

In this paper, we look in more detail at the interaction between the distinguishing features of *DL-Lite*$_F$ and *DL-Lite*$_R$, and we single out an extension of both *DL-Lite*$_F$ and *DL-Lite*$_R$ that is still LOGSPACE w.r.t. data complexity, and allows for delegating the data dependent part of the query answering process to to the relational DBMS managing the data layer. Moreover, we take seriously the distinction in OWL between objects and values (a distinction that typically is blurred in description logics), and introduce, besides concepts and roles, also concept-attributes and role-attributes, that describe properties of concepts (resp., roles) represented by values rather than objects. In fact, role attributes are currently not available in OWL, but are present in most conceptual modeling formalisms such as UML class diagrams and Entity-Relationship diagrams. Finally, we look at the problem of accessing databases that are independent from the ontology, and are related to the ontology through suitable mappings [9]. Observe, however, that such databases, being relational, store only values (not objects), and hence objects need to be constructed from such values, i.e., we have to deal with the so-called "impedance mismatch". We would like to highlight that the current work is one of the outcomes of the European TONES project[1] and that it complies with the TONES logical framework.

2 The description logic *DL-Lite*$_A$

In this section we present a new logic of the *DL-Lite* family, called *DL-Lite*$_A$. 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.

[1] `http://www.tonesproject.org/`

We first introduce the DL *DL-Lite*$_{FR}$, that combines the main features of two DLs presented in [4], called *DL-Lite*$_F$ and *DL-Lite*$_R$, respectively, and forms the basics of *DL-Lite*$_A$. In providing the specification of our logics, we use the following notation:

- A denotes an *atomic concept*, B a *basic concept*, and C a *general concept*;
- D denotes an *atomic value-domain*, E a *basic value-domain*, and F a *general value-domain*;
- P denotes an *atomic role*, Q a *basic role*, and R a *general role*;
- U_C denotes an *atomic concept attribute*, and V_C a *general concept attribute*;
- U_R denotes an *atomic role attribute*, and V_R a *general role attribute*;
- $\top_C$ denotes the *universal concept*, $\top_D$ denotes the *universal value-domain*.

Given a concept attribute U_C (resp. a role attribute U_R), we call the *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. Furthermore, we denote with $\delta_F(U_C)$ (resp. $\delta_F(U_R)$) the set of objects (resp. of pairs of objects) that U_C (resp. U_R) relates to values in the value-domain F. In particular, *DL-Lite*$_{FR}$ expressions are defined as follows.

- Concept expressions:

$$\begin{array}{l} B \ ::= \ A \mid \exists Q \mid \delta(U_C) \\ C \ ::= \ \top_C \mid B \mid \neg B \mid \exists Q.C \mid \delta_F(U_C) \mid \exists \delta_F(U_R) \mid \exists \delta_F(U_R)^- \end{array}$$

- Value-domain expressions (*rdfDataType* denotes predefined value-domains such as integers, strings, etc.):

$$\begin{array}{l} E \ ::= \ D \mid \rho(U_C) \mid \rho(U_R) \\ F \ ::= \ \top_D \mid E \mid \neg E \mid \mathit{rdfDataType} \end{array}$$

- Attribute expressions:

$$\begin{array}{l} V_C \ ::= \ U_C \mid \neg U_C \\ V_R \ ::= \ U_R \mid \neg U_R \end{array}$$

- Role expressions:

$$\begin{array}{l} Q \ ::= \ P \mid P^- \mid \delta(U_R) \mid \delta(U_R)^- \\ R \ ::= \ Q \mid \neg Q \mid \delta_F(U_R) \mid \delta_F(U_R)^- \end{array}$$

A *DL-Lite*$_{FR}$ *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*$_{FR}$ *TBox* assertions are of the form:

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

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.

As for the ABox, we introduce two disjoint alphabets, called Γ_O and Γ_V, respectively. Symbols in Γ_O, called object constants, are used to denote objects, while symbols in Γ_V, called value constants, are used to denote data values. A *DL-Lite*$_{FR}$ ABox is a finite set of assertions of the form:

$$A(a), \quad D(c), \quad P(a,b), \quad U_C(a,c), \quad U_R(a,b,c) \qquad \textit{membership assertions}$$

where a and b are constants in Γ_O, and c is a constant in Γ_V.

The semantics of *DL-Lite*$_{FR}$ is given in terms of FOL interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a first order structure over the interpretation domain $\Delta^{\mathcal{I}}$ that is the disjoint union of $\Delta_O^{\mathcal{I}}$ and $\Delta_V^{\mathcal{I}}$, with an *interpretation function* $\cdot^{\mathcal{I}}$ such that

- for all $a \in \Gamma_O$, we have that $a^{\mathcal{I}} \in \Delta_O^{\mathcal{I}}$;
- for all $a, b \in \Gamma_O$, we have that $a \neq b$ implies $a^{\mathcal{I}} \neq b^{\mathcal{I}}$;
- for all $c \in \Gamma_V$, we have that $c^{\mathcal{I}} \in \Delta_V^{\mathcal{I}}$;
- for all $c, d \in \Gamma_V$, we have that $c \neq d$ implies $c^{\mathcal{I}} \neq d^{\mathcal{I}}$;
- and the following conditions are satisfied (below, $o, o' \in \Delta_O^{\mathcal{I}}$, and $v \in \Delta_V^{\mathcal{I}}$; moreover, we do not report cases for $\delta(U_C)$ and $\delta(U_R)$ since they can be seen as abbreviations for $\delta_{\top_D}(U_C)$ and $\delta_{\top_D}(U_R)$, respectively):

$$\top_C^{\mathcal{I}} = \Delta_O^{\mathcal{I}} \qquad (P^-)^{\mathcal{I}} = \{\ (o,o') \mid (o',o) \in P^{\mathcal{I}}\ \}$$
$$\top_D^{\mathcal{I}} = \Delta_V^{\mathcal{I}} \qquad (\delta_F(U_C))^{\mathcal{I}} = \{\ o \mid \exists v.\, (o,v) \in U_C^{\mathcal{I}} \wedge v \in F^{\mathcal{I}}\ \}$$
$$A^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \qquad (\delta_F(U_R))^{\mathcal{I}} = \{\ (o,o') \mid \exists v.\, (o,o',v) \in U_R^{\mathcal{I}} \wedge v \in F^{\mathcal{I}}\ \}$$
$$D^{\mathcal{I}} \subseteq \Delta_V^{\mathcal{I}} \qquad (\delta_F(U_R)^-)^{\mathcal{I}} = \{\ (o,o') \mid \exists v.\, (o',o,v) \in U_R^{\mathcal{I}} \wedge v \in F^{\mathcal{I}}\ \}$$
$$P^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}} \qquad (\exists \delta_F(U_R))^{\mathcal{I}} = \{\ o \mid \exists\, o'.\, (o,o') \in (\delta_F(U_R))^{\mathcal{I}}\ \}$$
$$U_C^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}} \qquad (\exists \delta_F(U_R)^-)^{\mathcal{I}} = \{\ o \mid \exists\, o'.\, (o,o') \in (\delta_F(U_R)^-)^{\mathcal{I}}\ \}$$
$$U_R^{\mathcal{I}} \subseteq \Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}} \qquad (\exists Q)^{\mathcal{I}} = \{\ o \mid \exists o'.\, (o,o') \in Q^{\mathcal{I}}\ \}$$
$$(\neg U_C)^{\mathcal{I}} = (\Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus U_C^{\mathcal{I}} \qquad (\exists Q.C)^{\mathcal{I}} = \{\ o \mid \exists o'.\, (o,o') \in Q^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\ \}$$
$$(\neg U_R)^{\mathcal{I}} = (\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}) \setminus U_R^{\mathcal{I}} \qquad (\neg Q)^{\mathcal{I}} = (\Delta_O^{\mathcal{I}} \times \Delta_O^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$$
$$(\rho(U_C))^{\mathcal{I}} = \{\ v \mid \exists o.\, (o,v) \in U_C^{\mathcal{I}}\ \} \qquad (\neg B)^{\mathcal{I}} = \Delta_O^{\mathcal{I}} \setminus B^{\mathcal{I}}$$
$$(\rho(U_R))^{\mathcal{I}} = \{\ v \mid \exists o, o'.\, (o,o',v) \in U_R^{\mathcal{I}}\ \} \qquad (\neg E)^{\mathcal{I}} = \Delta_V^{\mathcal{I}} \setminus E^{\mathcal{I}}$$

We define when an interpretation $\mathcal{I}$ satisfies an assertion (i.e., is a model of the assertion) as follows (below, each e, possibly with subscript, is an element of either $\Delta_O^{\mathcal{I}}$ or $\Delta_V^{\mathcal{I}}$, depending on the context, each t, possibly with subscript, is a constant of either Γ_O or Γ_V, depending on the context, a and b are constants in Γ_O, and c is a constant in Γ_V). Specifically, an interpretation $\mathcal{I}$ *satisfies*:

- an inclusion assertion $\alpha \sqsubseteq \beta$, if $\alpha^{\mathcal{I}} \subseteq \beta^{\mathcal{I}}$;
- a functional assertion (funct γ), where γ is either P, P^-, or U_C, if, for each e_1, e_2, e_3, $(e_1, e_2) \in \gamma^{\mathcal{I}}$ and $(e_1, e_3) \in \gamma^{\mathcal{I}}$ implies $e_2 = e_3$;
- a functional assertion (funct U_R), if for each e_1, e_2, e_3, e_4, $(e_1, e_2, e_3) \in U_R^{\mathcal{I}}$ and $(e_1, e_2, e_4) \in U_R^{\mathcal{I}}$ implies $e_3 = e_4$;
- a membership assertion $\alpha(t)$, where α is either A or D, if $t^{\mathcal{I}} \in \alpha^{\mathcal{I}}$;
- a membership assertion $\beta(t_1, t_2)$, where β is either P or U_C, if $(t_1^{\mathcal{I}}, t_2^{\mathcal{I}}) \in \beta^{\mathcal{I}}$;
- a membership assertion $U_R(a, b, c)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}, c^{\mathcal{I}}) \in U_R^{\mathcal{I}}$.

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 KB $\mathcal{K}$ *logically implies* an assertion α if all models of $\mathcal{K}$ are also models of α.

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 $A(x_o)$, $P(x_o, y_o)$, $D(x_v)$, $U_C(x_o, x_v)$, or $U_R(x_o, y_o, x_v)$, where x_o, y_o are either variables in $\boldsymbol{x}$ and $\boldsymbol{y}$ (called *object variables*) or constants in Γ_O, whereas x_v is either a variable in $\boldsymbol{x}$ and $\boldsymbol{y}$ (called a *value variable*) or a constant in Γ_V. 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 $\boldsymbol{e} \in \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}$ such that, when we assign $\boldsymbol{e}$ 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* (for CQs and UCQs): given a knowledge base $\mathcal{K}$ and a query $q(\boldsymbol{x})$ over $\mathcal{K}$, return the *certain answers* to $q(\boldsymbol{x})$ over $\mathcal{K}$, i.e., all tuples $\boldsymbol{t}$ of elements of $\Gamma_V \cup \Gamma_O$ such that, when substituted to $\boldsymbol{x}$ in $q(\boldsymbol{x})$, we have that $\mathcal{K} \models q(\boldsymbol{t})$, i.e., such that $\boldsymbol{t}^{\mathcal{I}} \in q^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$.

From the results in [4] it follows that, in general, query answering over *DL-Lite*$_{FR}$ 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*$_{FR}$ 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 now introduce the DL *DL-Lite*$_A$, which differs from *DL-Lite*$_{FR}$ because it imposes a limitation on the use of the functionality assertions in the TBox. As we will discuss in Section 3, such limitation is sufficient to guarantee that query answering is in LOGSPACE w.r.t. data complexity, and thus it can be reduced to first-order query evaluation (see Section 3).

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

1. *for every atomic or inverse of an atomic role* Q *appearing in a concept of the form* $\exists Q.C$*, the assertions* $(\mathsf{funct}\ Q)$ *and* $(\mathsf{funct}\ Q^-)$ *are not in* $\mathcal{T}$*;*
2. *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* $\mathcal{T}$*;*
3. *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* $\mathcal{T}$*;*
4. *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*$_A$ 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. It can be shown that functionalities specified in a *DL-Lite*$_A$ TBox are not implicitly propagated in the TBox, and that this allows for LOGSPACE query answering.

Example. Let $\mathcal{T}$ be the TBox that models information about employees and projects through the following assertions (in the following, concept names are written in lowercase, role names are written in uppercase, attribute names are in boldface font, domain names are in Courier font):

tempEmp $\sqsubseteq$ *employee*	(1)	*tempEmp* $\sqsubseteq \exists\delta(\mathbf{until})$	(6)
manager $\sqsubseteq$ *employee*	(2)	*manager* $\sqsubseteq \exists$*MANAGES*	(7)
employee $\sqsubseteq \exists$*WORKS-FOR*.*project*	(3)	*MANAGES* $\sqsubseteq$ *WORKS-FOR*	(8)
$\delta(\mathbf{until}) \sqsubseteq$ *WORKS-FOR*	(4)	*manager* $\sqsubseteq \neg\exists\delta(\mathbf{until})$	(9)
(funct $\mathbf{until}$)	(5)	$\rho(\mathbf{until}) \sqsubseteq$ `xsd:date`	(10)

$\mathcal{T}$ states that: (1) every temporary employee is an employee; (2) every manager is an employee; (3) every employee works for at least one project; (4) the domain of the role attribute **until** is the role *WORKS-FOR*; (5) the role attribute **until** is functional; (6) every temporary employee *must* participate in a role having an associated role attribute **until** (such a role, by assertion (4), is the role *WORKS-FOR*); (7) every manager participates to the role *MANAGES*; (8) every instance of role *MANAGES* is an instance of the role *WORKS-FOR*; (9) no manager *can* participate in a role having an associated attribute **until**; (10) the range of the role attribute **until** is `xsd:date`. ∎

3 Query answering in *DL-Lite*$_A$

We discuss now reasoning in *DL-Lite*$_A$, and concentrate on the basic reasoning task in the context of using ontologies to access large data repositories, namely (conjunctive) query answering over a *DL-Lite*$_A$ knowledge base. The other forms of reasoning usual in DLs can actually be reduced to query answering (through reductions analogous to the ones reported in [3] for *DL-Lite*).

We now briefly sketch the technique for query answering which we have defined for *DL-Lite*$_A$. In a nutshell, the algorithm has a structure similar to the methods developed for the logics in the *DL-Lite* family [3]: in particular, query answering is basically reduced to evaluation of a first-order query over a relational database representing the ABox. Such first-order query is obtained by reformulating the original query based on the TBox assertions. For *DL-Lite*$_A$, this reformulation can be obtained through the query reformulation technique for *DLR-Lite*$_R$ defined in [4]. *DLR-Lite*$_R$ is a logic of the *DL-Lite* family which allows for expressing n-ary relations (but no functionality assertions). The possibility of reusing the query reformulation algorithm of *DLR-Lite*$_R$ is based on the fact that inclusion assertions in *DL-Lite*$_A$ can actually be expressed in *DLR-Lite*$_R$.

An important aspect which such a query answering strategy relies on is a *separation* property between different kinds of TBox assertions: in particular, TBox assertions are classified into: (*i*) *positive inclusion assertions (PIs)*, i.e., inclusions having a positive concept/role/concept attribute/role attribute on its right-hand side, (*ii*) *negative inclusion assertions (NIs)*, i.e., inclusions having a negated concept/role/concept attribute/role attribute on its right-hand side; (*iii*) *functionality assertions*, i.e., assertions of the form (funct φ), where φ is a role/inverse of a role/atomic concept attribute/atomic role attribute. Then, it can be shown that, after saturating the TBox, query answering can be done by considering in a separate way the set of PIs and the set of NIs and functionality assertions. More precisely, NIs and functionality assertions are relevant for the satisfiability of $\mathcal{K}$ (while PIs are not); moreover, in a satisfiable KB $\mathcal{K}$, only PIs are relevant for answering UCQs. We remark that the above separation property holds for *DL-Lite*$_A$, but it does not hold for *DL-Lite*$_{FR}$.

On the base of the query answering algorithm discussed above, we able to provide a LOGSPACE upper bound for the data complexity of answering UCQs in *DL-Lite*$_A$.

Theorem 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_A$ KB. Answering UCQs posed to $\mathcal{K}$ is in* LOGSPACE *with respect to data complexity.*

We point out that the assumption that the TBox $\mathcal{T}$ is expressed in *DL-Lite*$_A$ rather than in *DL-Lite*$_{FR}$ is essential for the above upper bound to hold: in fact, from the results in [4], it follows that, answering UCQs in *DL-Lite*$_{FR}$ is PTIME-hard with respect to data complexity. This implies that in general there exists no FOL reformulation q' of a UCQ q (only depending on $\mathcal{T}$) such that the certain answers to q in $\mathcal{K}$ correspond to the evaluation of q' in $\mathcal{A}$.

4 Linking data to *DL-Lite*$_A$ 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*$_A$. 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*$_A$, 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. While Γ_V contains data value constants as before, Γ_O is built starting from Γ_V and a set Λ of function symbols of any arity (possibly 0), as follows: If $f \in \Lambda$, the arity of f is n, and $d_1, \ldots, d_n \in \Gamma_V$, then $f(d_1, \ldots, d_n)$ is a term in Γ_O, called *object term*. In other words, object terms are either objects constants (i.e., function symbols of arity 0), or function symbols applied to data value constants. In the following we call Γ_T the subset of Γ_O constituted by object constants, i.e., object terms built with function symbols of arity 0. Also, we use Γ_{VT} to denote $\Gamma_V \cup \Gamma_T$.

To define the semantics of terms in Γ_O, we simply define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as before, and we observe that the interpretation function $\cdot^{\mathcal{I}}$ now assigns a different element of $\Delta^{\mathcal{I}}_O$ to every object term (not only object constant) in Γ_O (i.e., we enforce the unique name assumption also on object terms).

Let us now turn our attention to the problem of linking data in the sources to objects in the ontology. To this end, we assume that data sources have been wrapped into a relational database DB. Note that this assumption is indeed realistic, as many data federation tools that provide exactly this kind of service are currently available. In this way, we can assume that all relevant data are virtually represented and managed by a relational data engine. In particular, one consequence is that we can query our data by using SQL. Formally, we assume that the database DB is characterized by a relational schema, and the corresponding extension. In particular, for each relation schema R, DB contains a set of tuples whose values are taken from Γ_{VT}. Note that, by virtue of this assumption, DB may store both data value constants and object constants. The evaluation of an SQL query φ over a database DB, denoted $ans(\varphi, DB)$, returns the set of tuples (of the arity of φ) of elements of Γ_{VT} that satisfy φ in DB.

To realize the link, 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. Consider a *DL-Lite*$_A$ TBox in which *person* is a concept name, *CITY-OF-BIRTH* is a role name, $\mathbf{age}$ and $\mathbf{cityName}$ are concept attributes names, 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.

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

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 object terms (see below). ∎

The example shows that, in specifying mapping assertions, we need variable object terms, i.e., object terms containing variables. Indeed, we extend object terms to *variable object terms* by allowing also variables to appear in place of value constants.

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 ψ is a *DL-Lite*$_A$ conjunctive query without existential variables and without constants, but whose atoms may contain variable object terms, and φ is an SQL query, i.e., an open first-order formula, over the database DB.

We now describe the semantics of mapping assertions. To this end, we introduce the notion of ground instance of a formula. Let γ be a formula with free variables $\boldsymbol{x}$, and let $\boldsymbol{s}$ be a tuple of elements in Γ_{VT} of the same arity as $\boldsymbol{x}$. A ground instance $\gamma[\boldsymbol{x}/\boldsymbol{s}]$ of γ is obtained from γ by substituting every occurrence of x_i with s_i. We say that an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* the mapping assertion $\varphi \rightsquigarrow \psi$ wrt DB, if for every ground instance $\varphi[\boldsymbol{x}/\boldsymbol{s}] \rightsquigarrow \psi[\boldsymbol{x}/\boldsymbol{s}]$ of $\varphi \rightsquigarrow \psi$, we have that $ans(\varphi[\boldsymbol{x}/\boldsymbol{s}], DB) = true$ implies $\psi[\boldsymbol{x}/\boldsymbol{s}]^{\mathcal{I}} = true$ (where, for a ground atom $p(\boldsymbol{t})$, with $\boldsymbol{t} = (t_1, \ldots, t_n)$ a tuple of object-terms, we have that $p(\boldsymbol{t})^{\mathcal{I}} = true$ if $(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}}) \in p^{\mathcal{I}}$).

Finally, we can summarize the semantics of a *DL-Lite*$_A$ ontology with mapping assertions. Let DB be a database as defined above, $\mathcal{T}$ a *DL-Lite*$_A$ 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 $\langle \mathcal{T}, \mathcal{M}, DB \rangle$ 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 Section 2.

As we have seen in the previous section, both query answering and satisfiability in *DL-Lite*$_A$ reduce to query answering over an ABox seen as a database. In fact, the mappings define an ABox, which is the one obtained by instantiating the atoms in the conjunctive query on the right-hand side of mapping assertions with the constants retrieved by the SQL query on the left-hand side. Note that such constants may also instantiate variables appearing in variable object terms, giving rise to object terms in the atoms, and hence in the corresponding ABox. The object terms would not make any difference for satisfiability and query answering, and hence the above techniques still apply. However, we avoid to explicitely generate such an ABox, by proceeding as follows. 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, all variable object terms disappear, except those that are returned by q as instantiations of free variables. Hence, what we have obtained is a FOL query to the database whose answer then has to be processed in order to generate the object terms to be returned. Notice that no post-processing is necessary in the case where the query q does not contain any free object variable.

Example. 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:

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

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

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

Then, we unfold each atom with the corresponding left-hand side of the mapping query

$$q(z) \leftarrow \textit{S2}(n, s, \mathsf{Roma}), \textit{S1}(n, s, z),$$

where w is a new (existential) variable symbol. The obtained query can be then simply evaluated over the database in order to get the certain answers to q. ∎

5 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 (in particular query answering) that deals with the data. We have proposed one such fragment, *DL-Lite*$_A$, 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 the case in which binary roles are substituted by relations of arbitrary arity. We are currently implementing *DL-Lite*$_A$ on the QuOnto system[2] [1] (which originally was based on *DL-Lite*$_F$) to deal with all the features introduced in this paper: attributes, role and attribute inclusions, and mappings to pre-existing databases.

Acknowledgments

This research has been partially supported by the FET IST Specific Targeted Research Project "Thinking ONtologies (TONES) – FP6-7603 funded by the European Union under the 6th Framework Programme.

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*, 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.

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