MASTRO-I: Efficient integration of relational data through DL ontologies

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
calvanese@inf.unibz.it

[2] Dip. di Informatica e Sistemistica
SAPIENZA Università di Roma
lastname@dis.uniroma1.it

1 Introduction

The goal of data integration is to provide a uniform access to a set of heterogeneous data sources, freeing the user from the knowledge about where the data are, how they are stored, and how they can be accessed. The problem of designing effective data integration solutions has been addressed by several research and development projects in the last years. One of the outcomes of this research work is a clear conceptual architecture for data integration[1]. According to this architecture [9], the main components of a data integration system are the global schema, the sources, and the mapping. Thus, a data integration system is seen as a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{G}$ is the *global schema*, providing both a conceptual representation of the application domain, and a reconciled, integrated, and virtual view of the underlying sources.
- $\mathcal{S}$ is the *source schema*, i.e., the schema of the sources where real data are stored.
- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, constituted by a set of assertions establishing the connection between the elements of the global schema and those of the source schema. Two basic approaches have been proposed in the literature. The first approach, called *global-as-view* (or simply GAV), focuses on the elements of the global schema, and associates to each of them a view (query) over the sources. On the contrary, in the second approach, called *local-as-view* (or simply LAV), the focus is on the sources, in the sense that a view (query) over the global schema is associated to each of them.

We use the term "data integration management system" to denote a software tool supporting the conceptual architecture described above. Among the various services to be provided by a data integration management system, we concentrate on query answering: Queries are posed in terms of the global schema, and are to be answered by suitably reasoning on the global schema, and exploiting the mappings to access data at the sources.

Data integration is still one of the major challenges in Information Technology. One of the reasons is that large amounts of heterogeneous data are nowadays available within an organization, but these data have been often collected and stored by different

[1] Here we are concerned with the so-called centralized data integration. Other architectures, e.g. [4], are outside the scope of this paper.

applications and systems. Therefore, the need of accessing data by means of flexible and unified mechanisms is becoming more and more important. On the other hand, current commercial data integration tools have several drawbacks. In particular, none of them realizes the goal of describing the global schema independently from the sources. In particular, these tools do not allow for specifying integrity constraints in the global schema, and this implies that the global schema is a sort of data structure for accommodating a reconciled view of the source data, rather than a faithful description of the application domain. It follows that current state-of-the-art data integration tools do not support the conceptual architecture mentioned above.

In this paper, we present a comprehensive approach to, and a complete management system for ontology-based data integration. The system, called MASTRO-I, is based on the following principles:

- The system fully adheres to the conceptual architecture developed by the scientific community.
- The global schema is specified in terms of an ontology, specifically in terms of a TBox expressed in a tractable Description Logics, namely *DL-Lite*$_{\mathcal{A}}$. So, our approach conforms to the view that the global schema of a data integration system can be profitably represented by an ontology, so that clients can rely on a shared conceptualization when accessing the services provided by the system.
- The source schema is the schema of a relational database.
- The mapping language allows for expressing GAV *sound* mappings between the sources and the global schema. A GAV sound mapping specifies that the extension of a source view provides a subset of the tuples satisfying the corresponding element of the global schema.
  Moreover, the mapping language has specific mechanisms for addressing the so-called *impedance mismatch* problem. This problem arises from the fact that, while the data sources store values, the instances of concepts in the ontology (global schema) are objects, each one denoted by an identifier (e.g., a constant in logic), not to be confused with any data item.

MASTRO-I is based on the system QUONTO [1], a reasoner for *DL-Lite*$_{\mathcal{A}}$, and is coupled with DB2 Information Integrator, the IBM tool for data federation [2].

We point out that our proposal is not the first one advocating the use of ontologies in data integration (see, for example, [7, 2]). However, to the best of our knowledge, MASTRO-I is the first data integration management system addressing simultaneously the following aspects:

- providing a solution to the impedance mismatch problem;
- answering unions of conjunctive queries posed to the global schema according to a method which is sound and complete with respect to the semantics of the ontology;
- careful design of the various languages used in the system, resulting in a very efficient technique (LOGSPACE with respect to data complexity) which reduces query answering to standard SQL query evaluation over the sources.

One might wonder whether the expressive power of the data integration framework underlying MASTRO-I can be improved. We answer this question by showing

[2] `http://www-128.ibm.com/developerworks/db2/zones/db2ii/`

that even very slight extensions of the expressive abilities of MASTRO-I in modeling the three components of a data integration system lead beyond the LOGSPACE complexity bound.

We end this section by pointing out that MASTRO-I addresses the problem of data integration, and not the one of schema or ontology integration. In other words, MASTRO-I is not concerned with the task of building the ontology representing the global schema starting from the source schema, or from other ontologies. This task, which is strongly related to other important problems, such as database schema integration [3], ontology alignment, matching, merging, or integration, are outside the scope of MASTRO-I.

2 MASTRO-I: The data integration framework

In this section we instantiate the conceptual architecture for data integration systems introduced in Section 1, by describing the form of the global schema, the source schema, and the mapping for data integration systems managed by MASTRO-I.

The global schema. Global schemas managed by MASTRO-I are given in terms of TBoxes expressed in *DL-Lite*$_\mathcal{A}$ [5], a DL of the *DL-Lite* family. Besides the use of concepts and roles, denoting sets of objects and binary relations between objects, respectively, *DL-Lite*$_\mathcal{A}$ allows one to use value-domains, a.k.a. concrete domains, denoting unbounded sets of (data) values, and concept attributes, denoting binary relations between objects and values[3]. In particular, the value-domains that we consider here are those corresponding to unbounded (i.e., value-domains with an unbounded size) RDF data types, such as integers, real, strings, etc.

To describe *DL-Lite*$_\mathcal{A}$, we first introduce the DL *DL-Lite*$_{\mathcal{FR}}$, which combines the main features of two DLs presented in [6], called *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$, respectively. 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, $T_1, \ldots, T_n$ denote the n pairwise disjoint unbounded RDF data types used in our logic, and F denotes a *general value-domain*, which can be either an unbounded RDF data type T_i or the *universal value-domain* $\top_D$; 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*. Given an attribute U_C, we call *domain* of U_C, denoted by $\delta(U_C)$, the set of objects that U_C relates to values, and we call *range* of U_C, denoted by $\rho(U_C)$, the set of values related to objects by U_C.

We are now ready to define *DL-Lite*$_{\mathcal{FR}}$ expressions as follows.

– Basic and general concept expressions:

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

– Basic and general value-domain expressions:

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

[3] The logic discussed in [5] is actually more expressive than *DL-Lite*$_\mathcal{A}$, since it includes role attributes, user-defined domains, as well as inclusion assertions over such domains.

– General attribute expressions:

$$V_C \ ::= \ U_C \ \mid \ \neg U_C$$

– Basic and general role expressions:

$$Q ::= P \ \mid \ P^- \qquad R ::= Q \ \mid \ \neg Q$$

A *DL-Lite*$_{\mathcal{FR}}$ *TBox* allows one to represent intensional knowledge by means of assertions of the following forms:

– *Inclusion assertions*: $B \sqsubseteq C$ (concept inclusion assertion); $Q \sqsubseteq R$ (role inclusion assertion); $E \sqsubseteq F$ (value-domain inclusion assertion); $U_C \sqsubseteq V_C$ (attribute inclusion 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.
– *Functionality assertions* on atomic attributes or basic roles: $(\mathsf{funct}\ \ I)$, where I denotes either an atomic attribute or a basic role.

DL-Lite$_{\mathcal{A}}$ TBoxes are *DL-Lite*$_{\mathcal{FR}}$ TBoxes with suitable limitations in the combination of *DL-Lite*$_{\mathcal{FR}}$ TBox assertions. To describe such limitations we first introduce some preliminary notions. An atomic attribute U_C (resp. a basic role Q) is called an *identifying property in a TBox* $\mathcal{T}$, if $\mathcal{T}$ contains a functionality assertion $(\mathsf{funct}\ \ U_C)$ (resp. $(\mathsf{funct}\ \ Q)$ or $(\mathsf{funct}\ \ Q^-)$). Then, an atomic attribute or a basic role is called *primitive in* $\mathcal{T}$, if it does not appear positively in the right-hand side of an inclusion assertion of $\mathcal{T}$, and it does not appear in an expression of the form $\exists Q.C$ in $\mathcal{T}$.

Then, *a DL-Lite*$_{\mathcal{A}}$ *TBox is a DL-Lite*$_{\mathcal{FR}}$ *TBox* $\mathcal{T}$ *satisfying the condition that every identifying property in* $\mathcal{T}$ *is primitive in* $\mathcal{T}$.

Roughly speaking, in our logic, *identifying properties cannot be specialized*, i.e., they cannot be used positively in the right-hand side of inclusion assertions. As shown in [5], reasoning over a *DL-Lite*$_{\mathcal{A}}$ knowledge base (constituted by a TBox and an ABox) is tractable. More precisely, TBox reasoning is in PTIME and query answering is in LOGSPACE w.r.t. data complexity, i.e., the complexity measured in the size of the ABox only (whereas query answering for *DL-Lite*$_{\mathcal{FR}}$ is PTIME-hard). Thus, *DL-Lite*$_{\mathcal{A}}$ presents the same computational behavior of all DLs of the *DL-Lite* family, and therefore is particularly suited for integration of large amounts of data.

The source schema. The source schema in MASTRO-I is a flat relational database schema, representing the schemas of all the data sources. Since MASTRO-I integrates data sources that are distributed, possibly heterogeneous, and not necessarily in relational format, the source schema may in fact be obtained by wrapping a set of physical sources. Indeed, MASTRO-I is coupled with the IBM DB2 Information Integrator, and relies on both the wrapping facilities provided by this data federation tool, and on its ability to answer queries posed to a set of distributed physical sources.

The mapping. The mapping in MASTRO-I establishes the relationship between the source schema and the global schema, thus specifying how data stored at the sources are linked to the instances of the concepts and the roles in the global schema. To this aim, the mapping specification takes suitably into account the impedance mismatch problem, i.e., the mismatch between the way in which data is (and can be) represented in a data

source, and the way in which the corresponding information is rendered through the global schema.

The MASTRO-I mapping assertions keep data value constants separate from object identifiers, and construct identifiers as (logic) terms over data values. More precisely, object identifiers in MASTRO-I are *terms* of the form $f(d_1, \ldots, d_n)$, where f is a function symbol of arity $n > 0$, and $d_1, \ldots, d_n$ are data values stored at the sources. Note that this idea traces back to the work done in deductive object-oriented databases [8].

We detail below the above ideas. The mapping in MASTRO-I is a set of assertions of the following the forms:

- *Typing mapping assertions*: $\Phi(v) \rightsquigarrow T_i(v)$, where Φ is a query over the source schema $\mathcal{S}$ denoting the projection of one relation over one of its attributes, T_i is one of the *DL-Lite*$_\mathcal{A}$ data types, and v is a variable,
- *Data-to-ontology mapping assertions*: $\Phi(\boldsymbol{v}) \rightsquigarrow P(\boldsymbol{t}, \boldsymbol{v}')$, where Φ is a first-order logic (FOL) query over the source schema $\mathcal{S}$, P is an atom in the global schema $\mathcal{G}$, $\boldsymbol{v}, \boldsymbol{v}'$ are variables such that $\boldsymbol{v}' \subseteq \boldsymbol{v}$ and $\boldsymbol{t}$ are *variable object terms*, i.e., terms having the form $f(\boldsymbol{v}'')$, where f is a function symbol, and $\boldsymbol{v}''$ are variables such that $\boldsymbol{v}'' \subseteq \boldsymbol{v}$.

Typing mapping assertions are used to assign appropriate *DL-Lite*$_\mathcal{A}$ types to values occurring in the tuples at the sources. Basically, these assertions are used for interpreting the values stored at the sources in terms of the types used in the global schema. Data-to-ontology, on the other hand, are used to map source relations (and the tuples they store), to global concepts, roles, and attributes (and the objects and the values that constitute their instances).

3 MASTRO-I: Semantics

We now illustrate the semantics of a data integration system managed by MASTRO-I.

Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system. The general idea is to start with a database D for the source schema $\mathcal{S}$, i.e., the extensions of the data sources, and define the semantics of $\mathcal{J}$ as the set of intepretations for $\mathcal{G}$ that both satisfy the TBox assertions of $\mathcal{G}$, and satisfy the mapping assertions in $\mathcal{M}$ with respect to D.

The above informal definition makes use of different notions that we detail below.

- First, the notion of interpretation for $\mathcal{G}$ is the usual one in DL. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ for $\mathcal{G}$ consists of an interpretation domain $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$. $\Delta^\mathcal{I}$ is the disjoint union of the domain of objects $\Delta^\mathcal{I}_O$, and the domain of values $\Delta^\mathcal{I}_V$, while the interpretation function $\cdot^\mathcal{I}$ assigns the standard formal meaning to all expressions and assertions of the logic *DL-Lite*$_\mathcal{A}$ (see [5]). The only aspect which is not standard here is the need of dealing with objects denoted by terms (see previous section). To this end, we now introduce two disjoint alphabets, called Γ_O and Γ_V, respectively. Symbols in Γ_O are called object terms, and are used to denote objects, while symbols in Γ_V, called value constants, are used to denote data values. More precisely, 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 functional terms of arity 0, called object constants, or terms constituted

by a function symbol applied to data value constants. We are ready to state how the interpretation function $\cdot^{\mathcal{I}}$ treats Γ_V and Γ_O: $\cdot^{\mathcal{I}}$ simply assigns a different value in $\Delta_V^{\mathcal{I}}$ to each symbol in Γ_V, and a different element of $\Delta_O^{\mathcal{I}}$ to every object term (not only object constant) in Γ_O. In other words, *DL-Lite*$_{\mathcal{A}}$ enforces the unique name assumption on both value constants and object terms.

– To the aim of describing the semantics of mapping assertions with respect to a database D for the source schema $\mathcal{S}$, we first assume that all data values stored in the database D belong to Γ_V[4]. Then, if q is a query over the source schema $\mathcal{S}$, we denote by $ans(q, D)$ the set of tuples obtained by evaluating the query q over the database D (if q has not distinguished variables, then $ans(q, D)$ is a boolean). Finally, we introduce the notion of ground instance of a formula. Let γ be a formula with free variables $\boldsymbol{x} = (x_1, \ldots, x_n)$, and let $\boldsymbol{s} = (s_1, \ldots, s_n)$ be a tuple of elements in $\Gamma_V \cup \Gamma_O$. 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 are now ready to specify the semantics of mapping assertions. We say that an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* the mapping assertion $\varphi \rightsquigarrow \psi$ with respect to D, if for every ground instance $\varphi[\boldsymbol{x}/\boldsymbol{s}] \rightsquigarrow \psi[\boldsymbol{x}/\boldsymbol{s}]$ of $\varphi \rightsquigarrow \psi$, we have that $ans(\varphi[\boldsymbol{x}/\boldsymbol{s}], D) = true$ 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 $(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}}) \in p^{\mathcal{I}}$). Note that the above definition formalizes the notion of sound mapping, as it treats each mapping assertion as an implication.
– With the above notion in place, we can simply define the semantics of $\mathcal{J}$ with respect to D as follows:

$$sem_D(\mathcal{J}) = \{\, \mathcal{I} \mid \mathcal{I} \text{ is a model of } \mathcal{G}, \text{ and } \mathcal{I} \text{ satisfies all assertions in } \mathcal{M} \text{ wrt } D \,\}$$

As we said in the introduction, in this paper we are mainly interested in the problem of answering unions of conjunctive queries (UCQs) posed to the global schema. The semantics of query answering is given in terms of certain answers to the query, defined as follows. Given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and a database D for $\mathcal{S}$, the set of *certain answers* to the query $q(\boldsymbol{x})$ over $\mathcal{G}$ is the set (denoted by $ans(q, \mathcal{J}, D)$) of all tuples $\boldsymbol{t}$ of elements of $\Gamma_V \cup \Gamma_O$ such that $\mathcal{I} \models_{FOL} q[\boldsymbol{x}/\boldsymbol{t}]$ for every $\mathcal{I} \in sem_D(\mathcal{J})$ (notice that $q[\boldsymbol{x}/\boldsymbol{t}]$ is a boolean UCQ, i.e., a FOL sentence).

4 Query answering

In this section, we sketch our query answering technique (more details can be found in [10]). Consider a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a database D for $\mathcal{S}$.

We start with the following observation. Suppose we evaluate (over D) the queries in the left-hand sides of the mapping assertions, and we materialize accordingly the corresponding assertions in the right-hand sides. This would lead to a set of ground assertions, that can be considered as a *DL-Lite* ABox, denoted by $\mathcal{A}^{\mathcal{M},D}$. It can be shown that query answering over $\mathcal{J}$ can be reduced to query answering over the *DL-Lite*$_{\mathcal{A}}$ knowledge base constituted by the TBox $\mathcal{G}$ and the ABox $\mathcal{A}^{\mathcal{M},D}$. However, due to the materializion of $\mathcal{A}^{\mathcal{M},D}$, the query answering algorithm resulting from this approach

[4] We could also introduce suitable conversion functions in order to translate values stored at the sources into value constants in Γ_V, but we do not deal with this issue here.

would be polynomial in the size of D. On the contrary, our idea is to avoid any ABox materialization, but rather answer Q by reformulating it into a new query that can be afterwards evaluated directly over the database D. This can be achieved by following three steps, called *rewriting*, *unfolding* and *evaluation*.

Query rewriting. Given a UCQ Q over a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and a database D for $\mathcal{S}$, the rewriting step computes a UCQ Q' over $\mathcal{J}$, where the assertions of $\mathcal{G}$ are compiled in. It can be shown [10] that Q' is such that $ans(Q', \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle, D) = ans(Q, \mathcal{J}, D)$, i.e. rewriting allows to get rid of $\mathcal{G}$. Moreover, the rewriting procedure does not depend on D, runs in polynomial time in the size of $\mathcal{G}$, and returns a query Q' whose size is at most exponential in the size of Q.

Unfolding. Given a UCQ Q' over $\mathcal{J}$, this step computes, by using logic programming technology, an SQL query Q'' over the source schema $\mathcal{S}$, that possibly returns object terms. It can be shown [5, 10] that Q'' is such that $ans(Q'', D) = ans(Q', \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle, D)$, i.e. unfolding allows to get rid of $\mathcal{M}$. Moreover, the unfolding procedure does not depend on D, runs in polynomial time in the size of $\mathcal{M}$, and returns a query Q'', whose size is at most exponential in the size of Q'.

Evaluation. The evaluation step consists in simply delegating the evaluation of Q'' to the data federation tool managing the data sources. Formally, such a tool returns the set $ans(Q'', D,)$, i.e. the set of tuples obtained from the evaluation of Q'' over D.

From the above discussion, we have the following:

Theorem 1. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a* MASTRO-I *data integration system, and D a database for $\mathcal{S}$. Answering a UCQ over $\mathcal{J}$ with respect to D can be reduced to the evaluation of an SQL query over D, and is* LOGSPACE *in the size of D.*

Finally, we remark that we are implicitly assuming that the database D for $\mathcal{S}$ is consistent with the data integration system $\mathcal{J}$, i.e., $sem_D(\mathcal{J})$ is non-empty. Notably, checking consistency can also be reduced to sending appropriate SQL queries to the sources [5, 10].

5 Extending the data integration framework

In this section we study whether the data integration setting presented above can be extended while keeping the same complexity of query answering. In particular, we investigate possible extensions for all the three components $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ of the system.

Extensions to *DL-Lite*$_\mathcal{A}$. With regard to the logic used to express the global schema $\mathcal{G}$, the results in [6] already imply that it is not possible to go beyond *DL-Lite*$_\mathcal{A}$ (at least by means of the usual DL constructs) and at the same time keep the data complexity of query answering within LOGSPACE. Here we consider the possibility of removing the unique name assumption (UNA), i.e., the assumption that, in every intepretation of a data integration system $\mathcal{J}$, both two distinct value constants, and two distinct object terms denote two different domain elements. Unfortunately, this leads query answering out of LOGSPACE. This result can be proved by a reduction from Graph Reachability to instance checking in *DL-Lite*$_\mathcal{F}$ [6], i.e., query answering for a boolean query whose body is a single instantiated atom, over a DL that is a subset of *DL-Lite*$_\mathcal{A}$.

Theorem 2. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a* MASTRO-I *data integration system extended by removing the UNA, and D a database for $\mathcal{S}$. Answering a UCQ over $\mathcal{J}$ with respect to D is NLOGSPACE-hard in the size of D.*

Different source schemas. Although MASTRO-I is currently able to deal with relational sources only, it is not hard to see that all the results presented in this paper apply also if we consider data at the sources structured according to a different data model (e.g. XML). Obviously, depending on the specific data model, we have to resort to a suitable query language for expressing the source queries appearing in the mapping assertions. To adhere to our framework, the only constraint on this language is that it is able to extract tuples of values from the sources, a constraint that is trivially satisfied by virtually all query languages used in practice.

Extensions to the mapping language. As for the language used to express the mapping $\mathcal{M}$, we investigate the extension of the mapping language to allow for GLAV assertions, i.e., assertions that relate CQs over the sources to CQs over the global schema. Such assertions are therefore an extension of both GAV and LAV mappings. The result we present is that, even with LAV mappings only, instance checking and query answering are no more in LOGSPACE wrt data compexity.

Theorem 3. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a* MASTRO-I *data integration system extended with LAV mapping assertions, and D a database for $\mathcal{S}$. Answering a UCQ over $\mathcal{J}$ with respect to D is NLOGSPACE-hard in the size of D.*

The above result can be proved again by a reduction from Graph Reachability to instance checking in *DL-Lite*$_{\mathcal{F}}$.

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.
2. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-based integration of XML web resources. In *Proc. of ISWC 2002*, pages 117–131.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
4. P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Proc. of WebDB 2002*.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic DL-Lite$_A$. In *Proc. of OWLED 2006*.
6. 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.
7. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
8. R. Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases*, pages 193–256. Academic Press, 1988.
9. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246.
10. A. Poggi. *Structured and Semi-Structured Data Integration*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 2006.