MASTRO: A Reasoner for Effective Ontology-Based Data Access

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, Domenico Fabio Savo

Dip. di Ing. Informatica, Automatica e Sistemistica
Sapienza Università di Roma
lastname@dis.uniroma1.it

Abstract. In this paper we present MASTRO, a Java tool for ontology-based data access (OBDA) developed at Sapienza Università di Roma. MASTRO manages OBDA systems in which the ontology is specified in a logic of the *DL-Lite* family of Description Logics specifically tailored to ontology-based data access, and is connected to external data management systems through semantic mappings that associate SQL queries over the external data to the elements of the ontology. Advanced forms of integrity constraints, which turned out to be very useful in practical applications, are also enabled over the ontologies. Optimized algorithms for answering expressive queries are provided, as well as features for intensional reasoning and consistency checking. MASTRO has been successfully used in several projects carried out in collaboration with important organizations, on which we briefly comment in this paper.

1 Introduction

In this paper we present the current version of MASTRO, a system for ontology-based data access (OBDA) developed at Sapienza Università di Roma. MASTRO allows users for accessing external data sources through an ontology expressed in a fragment of the W3C Web Ontology Language (OWL).

As in data integration systems [11], mappings are used in OBDA to specify the semantic correspondence between a unified view of the domain (called global schema in data integration terminology) and the data stored at the sources. The distinguishing feature of the OBDA approach, however, is the fact that the global unified view is specified using an ontology language, which typically allows to provide a rather rich conceptualization of the domain of interest, that is independent from the representation adopted for the data stored at the sources. This choice provides several advantages: it allows for a declarative approach to data access and integration and provides a specification of the domain that is independent from the data layer; it realizes logical/physical independence of the information system, which is therefore more accessible to non-experts of the underlying databases; the conceptual approach to data access does not impose to fully integrate the data sources at once, as it often happens in data integration mediator-based system, but the design can be carried out in an incremental way;

the conceptual model available on the top of the system provides a common ground for the documentation of the data stores and can be seen as a formal specification for mediator design.

MASTRO has solid theoretical basis [3, 4]. In the current version of MASTRO, ontologies are specified in *DL-Lite*$_{A,id,den}$, a logic of the *DL-Lite family* of tractable Description Logics (DLs), which are specifically tailored to the management and querying of ontologies in which the extensional level, i.e., the data, largely dominates the intensional level. From the point of view of the expressive power, *DL-Lite*$_{A,id,den}$ captures the main modeling features of a variety of representation languages, such as basic ontology languages and conceptual data models. Furthermore, it allows for specifying advanced forms of identification constraints [5] and denials [10], that are not part of OWL 2, the current W3C standard language for specifying ontologies.

Answering unions of conjunctive queries in OBDA systems managed by MASTRO can be done through a very efficient technique that reduces this task to standard SQL query evaluation. Indeed, conjunctive query answering has been shown to be in LOGSPACE (in fact in AC^0) w.r.t. data complexity [4], i.e., the complexity measured only w.r.t. the extensional level, which is the same complexity of evaluating SQL queries over plain relational databases. One key feature of the current version of MASTRO, wrt previous ones [2], is that it adopts the *Presto* algorithm [15] for first-order query rewriting.

MASTRO is developed in Java and can be connected to any data management system allowing for a JDBC connection, e.g., a relational DBMS. In those cases in which several, possibly non-relational, sources need to be accessed, MASTRO can be coupled with a relational data federation tool[1], which wraps sources and represents them as a single (virtual) relational database.

The rest of the paper is organized as follows. In Section 2, we briefly describe the framework of ontology-based data access. In Section 3, we describe the query answering algorithm of the MASTRO system. In Section 4, we report on some real world information integration applications where MASTRO has been successfully trialed. In Section 5, we conclude the paper by discussing related work.

2 Ontology-based data access

In OBDA, the aim is to give users access to a data source or a collection thereof, by means of a high-level conceptual view specified as an ontology. The ontology is usually formalized in Description Logics (DLs) [1], which are at the basis of OWL. These logics allow one to represent the domain of interest in terms of *concepts*, denoting sets of objects (corresponding to OWL *classes*), *roles*, denoting binary relations between objects (OWL *object properties*), and *attributes*, denoting relations between objects and values from predefined domains (OWL *data properties*).

[1] E.g., IBM WebSphere Application Server (`http://www.ibm.com/software/webservers/appserv/was/`), Oracle Data Service Integrator (`http://www.oracle.com/us/products/middleware/data-integration/`).

A DL ontology is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ [1] where $\mathcal{T}$, called *TBox*, is a finite set of intensional assertions, and $\mathcal{A}$, called *ABox*, is a finite set of instance assertions, i.e, assertions on individuals. Different DLs allow for different kinds of TBox and/or ABox assertions.

The semantics of an ontology is given in terms of first-order interpretations [1]. An interpretation $\mathcal{I}$ is a *model* of an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it satisfies all assertions in $\mathcal{T} \cup \mathcal{A}$, where the notion of satisfaction depends on the constructs and axioms allowed by the specific DL in which $\mathcal{O}$ is expressed.

Among the extensional reasoning tasks w.r.t. a given ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, the most relevant ones are *ontology satisfiability* and *query answering*.

In particular, we are interested in the class of *conjunctive queries* (CQ). A CQ q over an ontology $\mathcal{O}$ (resp. TBox $\mathcal{T}$) 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 $A(z)$, $P(z, z')$, $U(z, z')$ where A is a concept name, P is a role name and U is an attribute name, and z, z' are either variables in $\boldsymbol{x}$ or in $\boldsymbol{y}$ or constants. The *arity* of q is the arity of $\boldsymbol{x}$. A CQ of arity 0 is called a *boolean conjunctive query*. A *union of conjunctive queries* (UCQ) is a query of the form $q(\boldsymbol{x}) \leftarrow \bigvee_i \exists \boldsymbol{y_i}.conj(\boldsymbol{x}, \boldsymbol{y_i})$.

Given a query $q(\boldsymbol{x})$ (either a conjunctive query or an union of conjunctive queries) and an ontology $\mathcal{O}$, the *certain answers* to $q(\boldsymbol{x})$ over $\mathcal{O}$ is the set $cert(q, \mathcal{O})$ of all tuples $\boldsymbol{t}$ of constants appearing in $\mathcal{O}$, such that, when substituted for the variables $\boldsymbol{x}$ in $q(\boldsymbol{x})$, we have that $\mathcal{O} \models q(\boldsymbol{t})$, meaning that $\boldsymbol{t}^{\mathcal{I}} \in q^{\mathcal{I}}$ for every $\mathcal{I} \in Mod(\mathcal{O})$. Notice that the answer to a boolean query is either the empty tuple, considered as $true$, or the empty set, considered as $false$.

In OBDA, the extensional level is not represented directly by an ABox, but rather by a database that is connected to the TBox by means of suitable mapping assertions[2]. Such *mapping assertions* have the form $\Phi \rightsquigarrow \Psi$, where Φ, called the *body* of the assertion, is an arbitrary SQL query over the underlying database, and Ψ, called the *head*, is a CQ over the TBox $\mathcal{T}$. Intuitively, a mapping assertion specifies that the tuples returned by the SQL query Φ are used to generate the facts that instantiate the concepts, roles, and attributes in Ψ.

All the notions given above can be easily generalized to OBDA systems, where a TBox $\mathcal{T}$ is connected to an external database $\mathcal{D}$ through mappings $\mathcal{M}$, denoted $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$. In particular, the *models* of $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ are those interpretations of $\mathcal{T}$ that satisfy the assertions in $\mathcal{T}$ and that are consistent with the tuples retrieved by $\mathcal{M}$ from $\mathcal{D}$ (see [13] for the formal details). Satisfiability amounts to checking whether $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ admits at least one model, while answering a query Q amounts to computing the tuples that are in the evaluation of Q in every model of $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$.

Mastro is able to deal with DL TBoxes that are expressed in *DL-Lite*$_{A,id,den}$, a member of the *DL-Lite* family of lightweight DLs [4]. In such DLs, a good tradeoff is achieved between the expressive power of the TBox lan-

[2] Note that, in the following, with some abuse of terminology, when we use the term "ontology" in the context of OBDA, we implicitly refer to the TBox only.

guage used to capture the domain semantics, and the computational complexity of inference, in particular when such a complexity is measured w.r.t. the size of the data.

Basic *DL-Lite*$_{A,id,den}$ expressions are defined as follows:

$$\begin{array}{llll} B \longrightarrow & A \mid \exists Q \mid \delta(U) & Q \longrightarrow & P \mid P^- \\ C \longrightarrow & B \mid \neg B & R \longrightarrow & Q \mid \neg Q \\ V \longrightarrow & U \mid \neg U & & \end{array} \qquad \begin{array}{ll} E \longrightarrow & \rho(U) \\ F \longrightarrow & T_1 \mid \cdots \mid T_n \end{array}$$

where, A, P, and P^- denote an *atomic concept*, an *atomic role*, and the *inverse of an atomic role* respectively; $\delta(U)$ (resp. $\rho(U)$) denotes the *domain* (resp. the *range*) of an *attribute* U, i.e., the set of objects (resp. values) that U relates to values (resp. objects); $T_1, \ldots, T_n$ are unbounded pairwise disjoint predefined value-domains; B is called *basic concept*.

A *DL-Lite*$_{A,id,den}$ TBox is a finite set of the following assertions:

$B \sqsubseteq C$	(*concept inclusion assertion*)
$Q \sqsubseteq R$	(*role inclusion assertion*)
$U \sqsubseteq V$	(*attribute inclusion assertion*)
$E \sqsubseteq F$	(*value-domain inclusion assertion*)
(funct Q)	(*role functionality assertion*)
(funct U)	(*attribute functionality assertion*)
(id $B\ \pi_1, \ldots, \pi_n$)	(*identification assertion*)
$\forall \boldsymbol{y}.conj(\boldsymbol{t}) \rightarrow \bot$	(*denial assertion*)

In identification assertions [5], π_i is a *path*, i.e., an expression built according to the following syntax: $\pi \longrightarrow S \mid D? \mid \pi_1 \circ \pi_2$, where S denotes an atomic role, the inverse of an atomic role, an attribute, or the inverse of an attribute, $\pi_1 \circ \pi_2$ denotes the composition of paths π_1 and π_2, and $D?$, called *test relation*, represents the identity relation on instances of D, which can be a basic concept or a value-domain. Test relations are used to impose that a path involves instances of a certain concept or value-domain. In *DL-Lite*$_{A,id,den}$, identification assertions are *local*, i.e., at least one $\pi_i \in \{\pi_1, ..., \pi_n\}$ has length 1, i.e., it is an atomic role, the inverse of an atomic role, or an attribute. Intuitively, an identification assertion of the above form asserts that for any two different instances o, o' of B, there is at least one π_i such that o and o' differ in the set of their π_i-fillers, that is the set of objects that are reachable from o by means of π_i.

In denial assertions [10], $conj(\boldsymbol{y})$ is defined as for boolean CQs. Intuitively, a denial assertion of the above form states that there must not exist any tuple $\boldsymbol{y}$ satisfying $conj(\boldsymbol{y})$, i.e., that the answer to the boolean query $q() \leftarrow \exists \boldsymbol{y}.conj(\boldsymbol{y})$ must be empty.

Finally, in a *DL-Lite*$_{A,id,den}$ TBox $\mathcal{T}$, the following condition must hold: each role or attribute that either is functional in $\mathcal{T}$ or appears (in either direct or inverse direction) in a path of an identification assertion in $\mathcal{T}$ is not specialized, i.e., it does not appear in the right-hand side of assertions of the form $Q \sqsubseteq Q'$ or $U \sqsubseteq U'$.

Mapping assertions handled by Mastro are assertions of the form $\Phi \rightsquigarrow \Psi$, where Φ is an arbitrary SQL query over the underlying database, and Ψ is a conjunction of atoms whose predicates are the concepts, roles, and attributes of the

TBox. Notice that, due to the fact that Ψ is a conjunction of atoms (as opposed to a query, possibly with existentially quantified variables), such mappings can be considered as a special form *global-as-view* (GAV) mappings [11].

In order to overcome the so-called *impedance mismatch* between the database, storing values, and the TBox, to be interpreted over a domain of objects, the mapping assertions are used in MASTRO to specify how to construct abstract objects from the tuples of values retrieved from the database. This is done by allowing one to use function symbols in the atoms in Ψ: together with the values retrieved by Φ, such function symbols generate so called *object terms*, which serve as object identifiers for individuals in the ontology. We notice that the semantics we adopt in MASTRO establishes that different terms denote different objects (unique name assumption), so that different terms never need to be equated during reasoning, which is coherent with the assumption of not having existentially quantified variables in the body of mappings.

For the logics of the *DL-Lite* family it has been shown that for unions of conjunctive queries (UCQs), under the unique name assumption, query answering can be carried out efficiently in the size of the data, by reducing it to SQL query evaluation over the ABox seen as a database [4]. Also satisfiability, which is easily reducible to query answering, can be solved through the same mechanism. Such techniques are implemented in MASTRO, we refer to [4, 13] for a more complete treatment.

As an example, consider the OBDA system $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$, where the TBox $\mathcal{T}$ is constituted by the following set of intensional assertions: $\{NationalFlight \sqsubseteq Flight$, $InternationalFlight \sqsubseteq Flight\}$, $\mathcal{D}$ is a database constituted by a set of relations with the following signature:

```
FL_TB[fl_num:string, departure:integer, arrival:integer],
AIRPORT_TB[airpt_code:integer, name:string, country:string],
```

and $\mathcal{M}$ contains the following mapping assertions:

```
SELECT fl_num
FROM FL_TB,AIRPORT_TB A1,AIRPORT_TB A2
WHERE departure = A1.airpt_code and
arrival = A2.airpt_code and
A1.country = 'IT' and A2.country = 'IT'
```

$\rightsquigarrow$ *NationalFlight*(**fl**(`fl_num`))

```
SELECT fl_num
FROM FL_TB,AIRPORT_TB A1,AIRPORT_TB A2
WHERE departure = A1.airpt_code and
arrival = A2.airpt_code and
(A1.country != 'IT' or A2.country != 'IT')
```

$\rightsquigarrow$ *InternationalFlight*(**fl**(`fl_num`))

which specify how to construct instances of the ontology concepts *NationalFlight* and *InternationalFlight* starting from the database relations `FL_TB` and `AIRPORT_TB`.

3 Query Answering

In this section we describe the query rewriting process of the MASTRO system. The technique is purely intensional and is performed in three steps (see Figure 1):

1. *TBox rewriting:* The first step rewrites the input UCQ according to the knowledge expressed by the TBox. The rewriting, performed using the Presto algorithm [15], produces as output a non-recursive Datalog program, which encodes the knowledge expressed by the TBox and the user query. The output Datalog program contains the definition of auxiliary predicates, not belonging to the alphabet of the ontology.
2. *Datalog Unfolding:* The output of the first step is then unfolded into a new UCQ by means of the *Datalog Unfolding* algorithm. It consists of a classic rule unfolding technique which eliminates all the auxiliary predicate symbols introduced by the Presto algorithm and produces a final UCQ expressed in terms of ontology concepts, roles, and attributes.
3. *Mapping Unfolding:* The last step takes the unfolded UCQ and the mapping assertions as input and produces an SQL query which can be directly evaluated over the data sources. In particular, the mapping assertions are first *split* into assertions of a simpler form, in which the head of every mapping assertion contains only a single ontology predicate; then, the final reformulation is produced through a mapping unfolding step, as described in [13].

More specifically, the Presto algorithm is an optimization of the well-known *PerfectRef* [4]. The latter, depending on the particular TBox being used, may lead to huge UCQs, consisting of many possibly redundant queries which can be eliminated from the final result. Presto tries to overcome such issue, rewriting the user query into a Datalog program whose rules encode only necessary expansion steps, thus preventing the generation of useless queries. It is important to note that after the Datalog unfolding program, one can have again an exponential number of queries, but MASTRO experiences on real world application showed a dramatic performance improvement w.r.t. to the performance of *PerfectRef*.

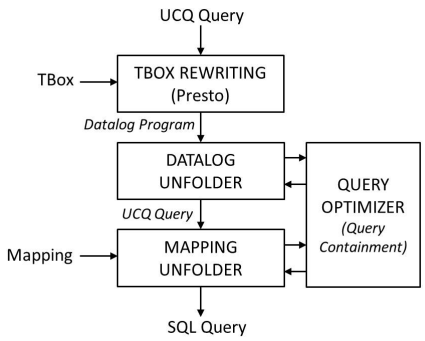

Fig. 1. The MASTRO rewriting process

4 The system at work: experiences on real cases

The usefulness of OBDA and the efficiency of the MASTRO system were proved by several real world applications in which it has been experimented. In the following, we report on the experiments carried out with Banca Monte dei Paschi di Siena (MPS), the Italian Ministry of Economy and Finance (MEF) and the Telecom Italia, the main Italian telephone company. Other experiments have been recently carried out with SELEX Sistemi Integrati (SELEX-SI), and Accenture [2].

Monte dei Paschi di Siena. Within a joint project with Banca Monte dei Paschi di Siena (MPS)[3], Free University of Bozen-Bolzano, and Sapienza Università di Roma, we used MASTRO for accessing a set of data sources from the actual MPS data repository by means of an ontology [16]. In particular, we focused on the data exploited by MPS personnel for risk estimation in the process of granting credit to bank customers. A 15 million tuple database, stored in 12 relational tables managed by the IBM DB2 RDBMS, has been used as data source collection in the experimentation. Such source data are managed by a dedicated application, which is in charge of guaranteeing data integrity (in fact, the underlying database does not force constraints on data). Not only the application performs various updates, but data is updated on a daily basis to identify connections between customers that are relevant for the credit rating estimation.

The main challenge that we tackled within the experimentation was the ontology and mapping design. This was a seven man-months process that required to both inspect the data source and interview domain experts, and was complicated by the fact that the source was managed by a specific application. The resulting OBDA system is defined in terms of approximately 600 $DL\text{-}Lite_{A,id}$ assertions over 79 concepts, 33 roles and 37 attributes, and 200 mapping assertions.

The experimentation showed that the usefulness of the MASTRO system goes beyond data integration applications and embraces data quality management. In particular, it confirmed the importance of several distinguished features of our system, namely, identification constraints and denial constraints, which have been used extensively to model important business rules. Notably, checking that such rules were satisfied by data retrieved from the sources through mappings led to highlight unexpected incompleteness and inconsistency in the data sources.

Our work has also pointed out the importance of the ontology itself, as a precious documentation tool for the organization. Indeed, the ontology developed in our project is adopted in MPS as a specification of the relevant concepts in the organization. At present we are still working with MPS in order to extend the work to cover the core domain of the MPS information system, with the idea that the ontology-based approach could result in a basic step for the future IT architecture evolution.

[3] MPS is one of the main banks, and the head company of the third banking group in Italy (see `http://english.mps.it/`).

Italian Ministry of Economy and Finance. MASTRO has been used within a joint project between Sapienza Università di Roma and the Italian Ministry of Economy and Finance (MEF). The main objectives of the project have been: the design and specification in *DL-Lite*$_A$ of an ontology for the domain of the Italian public debt; the realization of the mapping between the ontology and relational data sources that are part of the management accounting system currently in use at the ministry; the definition and execution of queries over the ontology aimed at extracting data of core interest for MEF users. In particular, the information returned by such queries relates to sales of bonds issued by the Italian government, maturities of bonds, monitoring of various financial products, etc., and are at the basis of various reports on the overall trend of the national public debt.

The Italian public dept ontology is over an alphabet containing 164 atomic concepts, 47 atomic roles, 86 attributes, and comprises around 1440 *DL-Lite*$_A$ assertions. The 300 mapping assertions involve around 60 relational tables managed by Microsoft SQLServer. We tested a very high number of queries and produced through MASTRO several reports of interest for the ministry. We point out that around 80% of the queries we tested could be executed only thanks to a series of further optimizations introduced in the system that, due to lack of space, we cannot describe here.

Telecom Italia. We finally describe a project we are carrying out in the domain of network inventory systems, together with Telecom Italia, the main Italian company for telecommunication services, which is also a world leading company in this field. The main objectives of the project are (*i*) the specification of an ontology that formalizes the entire telecommunication network owned by Telecom Italia and (*ii*) the analysis through the ontology of the information systems that are currently used for network management. The ontology we are going to develop can be partitioned into four layers: *Infrastructures and territory layer*, which represents main infrastructures used to realize the network, the way in which network elements (e.g., cables, apparatus, connection points) are localized into such infrastructures, and how both infrastructures and network elements are localized with respect to the territory; *network topology layer*, which represents how connections are realized into the network, essentially representing it as a graph in which edges represent elementary connections among apparatus, and nodes represent apparatus which realizes signal permutations between elementary connections; *service layer,* which represents all telecommunication services that are deployed on the network and offered to customer (e.g., voice communication, ADSL, voip); *data layer*, which represents the actual data exchanged on the network (e.g., data on telephone calls, internet access). In each such layer, the ontology provides the means to precisely represent the current state of the world, and, when considered of interest, also captures past situations, for example to provide tracks to all changes to which certain in the network have undergone. The use of identification assertions and epistemic constraints turned out to be crucial for faithful representation of such aspects.

5 Discussion

Accessing (possibly disperse) data through a virtual global schema has been deeply investigated in the last two decades in the field of data integration [11, 8]. From the modeling perspective, however, the main systems produced by this research suffer from some weakness, mainly due to the limited expressive power of the languages provided to model the global schema of the integration system. In this respect, Mastro aims at overcoming this limitation by providing the best expressive power allowed while preserving tractability of conjunctive query answering and of the integration tasks. As for the mappings, Mastro adopts a powerful form of the so-called Global-As-View (GAV) mappings [11], and provides optimized algorithms for rewriting global queries with respect their specification.

To the best of our knowledge, the only existing system designed for the same aims of Mastro is Quest [14], which has indeed common roots with our tool. Quest is a system for query answering over *DL-Lite*$_A$ ontologies, which can work in both "classical" (i.e., with a local ABox) and "virtual" mode (i.e., as an OBDA system). Quest implements specific optimizations for query answering, which in particular exploit completeness of the ABox with respect to the TBox. Although first experiments show effectiveness of Quest in the classical scenario [14], its usage in the virtual mode is in a still preliminary stage. In particular, we tried to compare Quest with Mastro in the OBDA scenario of the Italian Ministry of Economy and Finance described described in the previous section. Unfortunately, we have not been able to perform such experiments for two reasons: (i) the data source of this application is an SQL Server database; since Quest does not support this DBMS, we could not compare query answering in the two systems; (ii) Quest was not actually able to compute the TBox rewriting of the 23 queries used in our experiments, which are very long conjunctions of atoms, so we could not even compare the query rewriting performances of the two systems.

Nyaya [6] is a novel system which allows for query answering over ontologies specified into linear Datalog$^{\pm}$, a language that essentially corresponds to *DLR-Lite* [3] (i.e., to the extension of *DL-Lite* with n-ary predicates), and allows for FOL-rewritable query answering of UCQs. In Nyaya, Datalog$^{\pm}$ ontologies are mapped through plain Datalog rules to a specific centralized storage system which maintains both data and meta-data according to the Nyaya meta-model. As in Mastro, answering a query posed over such an ontology is done by first rewriting the query according to the ontology, using an algorithm which can be seen as a variation of the PerfectRef algorithm of [4], and then rewriting it according to the mapping, which is done in Nyaya through standard unfolding. Nyaya does not present particular optimizations for both the rewriting steps, whereas it concentrates in optimizing centralized data storage. In this respect, it is not specifically tailored to data integration and cannot be directly applied in an efficient way to this setting.

Other *DL-Lite*-based approaches and reasoners have been developed, which, however, are not able to deal with full OBDA scenarios. In [9] an alternative approach to query answering is presented. Besides a (less complex) query reformu-

lation step, such an approach requires to suitably "extend" the ABox (managed by a RDBMS) with the aim of reducing the amount of rewritten queries produced by the reformulation step. The experimental results support well this approach (notice that in MASTRO the size of the reformulation may be exponential in the size of the input query). However, the ABox manipulation that it requires makes it extremely difficult to apply this approach in an OBDA scenario.

The REQUIEM reasoner [12] implements a rewriting algorithm which reduces the number of queries in the final reformulation, still being purely intensional like MASTRO. However, it currently supports none of the MASTRO advanced features, such as identification or EQL constraint management, nor mappings to external databases.

The OWLGres prototype [19], which allows for TBox specification in *DL-Lite*, uses the PostgreSQL DBMS for the storage of the ABox, and provides conjunctive query processing. The algorithm for query answering implemented in OWLGres, however, is not complete with respect to the computation of the certain answers to user queries.

MASTRO can also be compared with ontology reasoners which support DLs different from *DL-Lite*, and in particular with their query answering capabilities. In this respect, well-known DL reasoners such as RacerPro [7], Pellet [18], Fact++ [20], and HermiT [17] provide only limited forms of query answering, i.e., instance checking/retrieval or *grounded* conjunctive query answering (c.f. [2]), since they are essentially focused on standard DL reasoning services. Although some optimizations have been implemented, such systems are not able to deal with very large ABoxes (e.g., with several millions of membership assertions) as the ones we considered in our experiments. This is mainly due to the inherent computational complexity of answering queries in the expressive DL languages supported by the above mentioned systems.

Acknowledgments. This research has been partially supported by the EU under FP7 project ACSI – Artifact-Centric Service Interoperation (grant n. FP7-257593), and by Regione Lazio under the project "Integrazione semantica di dati e servizi per le aziende in rete".

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The Mastro system for ontology-based data access. *Semantic Web J.*, 2(1):43–53, 2011.
3. 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.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Path-based identification constraints in description logics. In *Proc. of KR 2008*, pages 231–241, 2008.
6. R. de Virgilio, G. Orsi, L. Tanca, and R. Torlone. Semantic data markets: a flexible environment for knowledge management. In *Proc. of CIKM 2011*, pages 1559–1564, 2011.
7. V. Haarslev, R. Möller, and M. Wessel. Description logic inference technology: Lessions learned in the trenches. In *Proc. of DL 2005*, volume 147 of *CEUR,* `ceur-ws.org`, 2005.
8. A. Y. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proc. of VLDB 2006*, pages 9–16, 2006.
9. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of KR 2010*, pages 247–257, 2010.
10. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant first-order rewritability of dl-lite with identification and denial assertions. In *Proc. of DL 2012*, volume 846 of *CEUR,* `ceur-ws.org`, 2012.
11. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.
12. H. Pérez-Urbina, B. Motik, and I. Horrocks. A comparison of query rewriting techniques for *DL-lite*. In *Proc. of DL 2009*, volume 477 of *CEUR,* `ceur-ws.org`, 2009.
13. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
14. M. Rodriguez-Muro and D. Calvanese. High performance query answering over dl-lite ontologies. In *Proc. of KR 2012*, 2012. To appear.
15. R. Rosati and A. Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proc. of KR 2010*, pages 290–300, 2010.
16. D. F. Savo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, V. Romagnoli, M. Ruzzi, and G. Stella. MASTRO at work: Experiences on ontology-based data access. In *Proc. of DL 2010*, volume 573 of *CEUR,* `ceur-ws.org`, pages 20–31, 2010.
17. R. Shearer, B. Motik, and I. Horrocks. HermiT: A highly-efficient OWL reasoner. In *Proc. of OWLED 2008*, volume 432 of *CEUR,* `ceur-ws.org`, 2008.
18. E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of DL 2004*, volume 104 of *CEUR,* `ceur-ws.org`, 2004.
19. M. Stocker and M. Smith. Owlgres: A scalable OWL reasoner. In *Proc. of OWLED 2008*, volume 432 of *CEUR,* `ceur-ws.org`, 2008.
20. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of IJCAR 2006*, pages 292–297, 2006.